November 2015

Indoor And Outdoor Real Time Information Collection in Disaster Scenario

Dongyi Yang
University of Massachusetts Amherst

INDOOR AND OUTDOOR REAL TIME INFORMATION COLLECTION IN DISASTER SCENARIO

A Thesis Presented

by

DONGYI YANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2015

Department of Electrical and Computer Engineering

INDOOR AND OUTDOOR REAL TIME INFORMATION COLLECTION IN DISASTER SCENARIO

A Thesis Presented

by

DONGYI YANG

Approved as to style and content by:

_____
Aura Ganz, Chair

_____
C. Mani Krishna, Member

_____
Russell Tessier, Member

_____
Christopher V. Hollot, Department Head
Department of Electrical and Computer Engineering

ABSTRACT

INDOOR AND OUTDOOR REAL TIME INFORMATION COLLECTION IN DISASTER SCENARIO

SEPTEMBER 2015

DONGYI YANG, B.S., DONGHUA UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Aura Ganz

A disaster usually severely harms human health and property. After a disaster, great amount of information of a disaster area is needed urgently. The information not only indicates the severity of the disaster, but also is crucial for an efficient search and rescue process. In order to quickly and accurately collect real time information in a disaster scenario, a mobile platform is developed for an outdoor scenario and a localization and navigation system for responders is introduced for an indoor scenario.

The mobile platform has been integrated to the DIORAMA system. It is built with a 6-wheel robot chassis along with an Arduino microcontroller. Controlled by a mounted Android smartphone, the mobile platform can receive commands from incident commanders and quickly respond to the commands. While patrolling in a disaster area, a constant RFID signal is collected to improve the localization accuracy of victims. Pictures and videos are also captured in order to enhance the situational awareness of rescuers.

The design of the indoor information collection is focused on the responder side. During a disaster scenario, it is hard to track responders' locations in an indoor environment. In this thesis, an indoor localization and navigation system based on Bluetooth low energy and Android is developed for helping responders report current location and quickly find the right path in the environment. Different localization algorithms are investigated and implemented. A navigation system based on A^* is also proposed.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Mass Casualty Incident

A mass casualty incident (often shortened to MCI and sometimes called a multiple-casualty incident or multiple-casualty situation) is any incident in which emergency medical services resources, such as personnel and equipment, are overwhelmed by the number and severity of casualties. [1]

Types of incidents that can produce mass casualties include, but are not limited to:

- Multiple vehicle collision
- Building collapse
- Mass transit accident
- CO emergency
- Hazard material incident
- Weapon of mass destruction
- Chemical exposure

1.2 DIORAMA

Effectively handling a mass-casualty incident is one of the greatest challenges a community's emergency medical system can face. Disasters disrupt existing infrastructures and can hamper the efficiency of search-and-rescue units. In a disaster medicine triage, determining the number and location of victims, relative to the location of available resources is crucial for maximizing and hastening rescue efforts.

Currently there are many MCI management systems, which implement a number of high-tech methods to help the rescue process. One of them is the Dynamic Information Collection and Resource Tracking System for Disaster Management (DIORAMA) system, developed by 5G

Mobile Evolution Laboratory in the University of Massachusetts Amherst [2, 3]. The DIORAMA system could provide the following capabilities:

- Real-time, scalable decision-support framework built for rapid information collection
- Accurate resource tracking
- Ability to identify the location and status of the casualties, responders, and emergency transport vehicles involved in a mass-casualty incident.
- Provide a visual aid that identifies the location and condition of the casualties as well as the available resources.

In DIORAMA, the localizations of patients are obtained by tagging patients with active Radio-Frequency Identification (RFID) tags. Each responder carries an Android Smartphone and an active RFID reader. The active RFID reader collects signal strength readings of the active tags. These readings along with the responders' GPS coordinates are sent through 4G networks to the DIORAMA server, which computes the location of each responder. Since the localization infrastructure is mobile (active RFID readers are carried by the responders), the localization is opportunistic, i.e., the localization can occur only if the responder is at a reasonable distance from the patient (closer than 20 meters). An incident commander with an Android tablet could monitor the whole rescue process in real time, getting information of responders' locations, victims' locations and priorities as well as a map of the incident area.

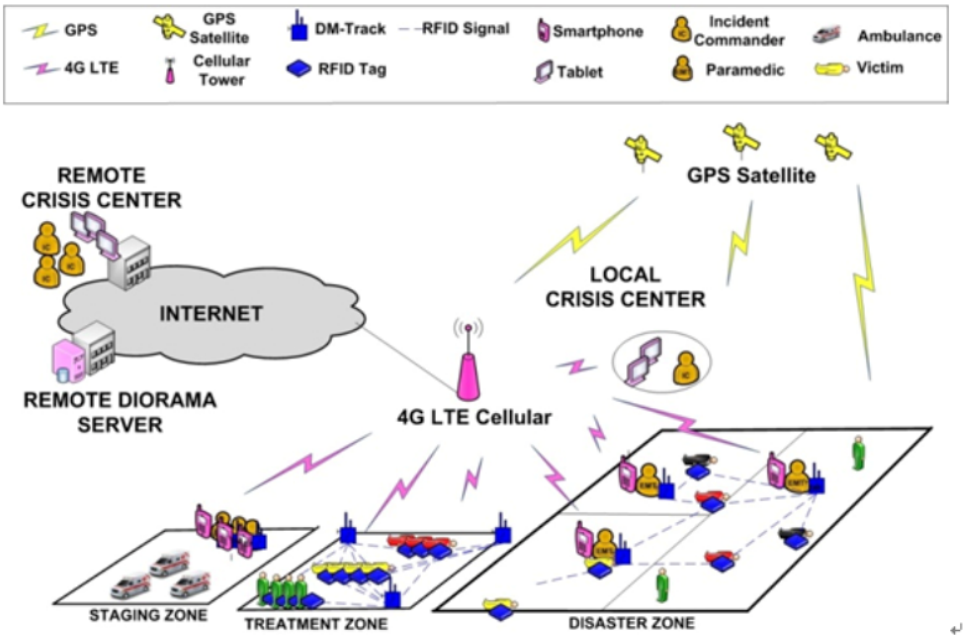


Figure.1. DIORAMA System Architecture

1.3 Mobile Platform for Outdoor Information Collection

A robot mobile platform could offer great help in the DIORAMA system. As the DIORAMA system uses RSSI information to localize victims, a RF reader could be mounted on the robot platform to receive RF signals. The RSSI information collected by the robot can increase the likelihood of getting the calculated location for victims. As a result, the localization information generated by the DIORAMA system is more accurate and more believable.

Secondly, as the localization infrastructure is mobile and the localization is opportunistic, the result of localization might be inaccurate. As Figure.2(a) shows, with the restriction of personnel resources, the actual location of one critical victim is not be covered by any responders. What's more, victim might move him/herself during the search and rescue process. If a victim moves out of the coverage of responders, an inaccurate localization can also be generated. So neither the incident commander nor responders would know his/her up-to-date location.

This issue can be solved by integrating an autonomous mobile robot into the system. As Figure.3(b) shows, the robot has the same covering range as a human responder. As responders move, a robot can be sent to areas that are not covered. A cooperating algorithm can be designed for the robot to cover those victims in low confidence areas as well as mobile victims.

Furthermore, with the camera equipped on the mobile platform, real time videos and pictures can be taken. This gives the incident commander as well as human responders a better understanding of the disaster environment.

(a) RF reading Coverage by Responder. One victim is not covered.

(b) Cooperation of Robot and Responders.
Figure.2. RF Information Collection

1.4 Indoor Search and Rescue

The current DIORAMA system mainly focuses on an outdoor scenario. The real time information collection of an indoor disaster scenario is even more important. Every year, a great number of tragedies happen during an indoor disaster search and rescue process due to a lack of

information. For example, in the Worcester cold storage and warehouse fire in 1999, 6 firefighters perished as a result of getting lost and disoriented in the building. So in this thesis, the real time information collection for localizing and navigating responders in an indoor environment is also studied.

The most important and useful information is the locations of responders. For an outdoor scenario, GPS is used to provide the Location Based Service (LBS). Unfortunately, when it comes to an indoor search and rescue scenario, GPS is not a good option. GPS requires line-of-sight (LOS) signals between satellites and a receiver. But in an indoor environment, satellite signals are attenuated and scattered greatly by roofs, walls, and other objects, making the accuracy of GPS unacceptable [4].

One of the best alternatives to GPS in an indoor environment is RF-based technology. A number of localization methods have been proposed using Wi-Fi, Radio Frequency Identification (RFID) and Bluetooth technology. All of these techniques use the 2.4 GHz industrial, scientific and medical (ISM) band [5]. Among Wi-Fi, RFID and Bluetooth, Bluetooth has many advantages over the other two. Nowadays smartphones, tablets and computers are normally equipped with standard Bluetooth chips. What is more, instead of purchasing expensive Wi-Fi access point or RFID receiver, the cost of a Bluetooth chip is much less. With the development of the Bluetooth Low Energy (BLE) technology, the maintenance cost for a BLE-based indoor localization system is also much less. A single coin battery can power a BLE device operates for years. The size of a BLE device is usually small, which makes it a perfect candidate for fast deployment in a new indoor environment during indoor search and rescue.

My contribution for the indoor real time information collection is designing and implementing the indoor localization and navigation system for human responders, including the indoor radio propagation model, beacon deployment, localization algorithms and navigation.

CHAPTER 2

LITERATURE SURVEY

There have been a number of research projects for both outdoor and indoor search and rescue process. A brief literature survey for both robot-integrated disaster response systems and indoor search and rescue systems is presented in this chapter.

2.1. Literature Survey on Robot-Integrated Search and Rescue System

The use of robot in a disaster search and rescue process has caused great interest among researches. In [6], researches introduced multiple robots into the search and rescue process and a coordination procedure is proposed for robots to perform real-time explorations over disaster areas. In the paper, every robot must maintain a valid communication channel with a human commander who is responsible for monitoring all the robots. The communication network is based on an ad hoc network. So the robots are autonomously classified into two kinds: communication relay robots and search robots. Computer simulations are executed in a virtual disaster area based on their method.

In [7], the author proposed autonomous robotic strategies for urban search and rescue (USAR), which are map-based semi-autonomous robot navigation and fully autonomous robotic search, tracking, localization and mapping (STLAM) using a team of robots. The so-called grid-based scan-to-map matching is proposed to correct robot estimation error, such as orientation error and localization error.

In [8], the authors developed a robot-integrated system called SENEKA, the objective of which is to network various robots and sensor systems used by first responders in order to make the search for victims and survivors quicker and more efficient. The project contains popular topics such as collision-free path planning for UAV/UGV, 3-D mapping by LIDAR and high-resolution 2-D mapping of a disaster area. Varies sensors are also proposed to help search and detect victims.

While in [9], the authors applied a wireless sensor network to disaster management. A

robot-integrated disaster management system is proposed. In the system both human rescuers and robots are equipped with Xbee-Pro board. So that two fixed nodes can measure the RSSI of each node and provide localization information. With RSSI information, robots can also cooperate with human rescuers and cover most of the disaster area. The robots are also equipped with air quality sensors and human detection modules to search and localize victims.

Also in [10], it presents a novel approach of using autonomous mobile robots to deploy a Wireless Sensor Network (WSN) for human existence detection in case of disasters. In their system, multiple robots are also classified into relay robot and explore robot. The Robots acting as explorers explore an unknown region by using frontier-based exploration strategy and communicate their findings to a relay in previously determined rendezvous points. The relays maintain connection between the base station and explorers. This strategy offers a solution to connectivity related problems at the expense of additional robots responsible for messaging. Besides building WSN communication, these robots are also used for multi-robot SLAM.

In [11], a novel communication scheme of an autonomous robot team via Bluetooth radio is investigated. In the presented solution, an autonomous unit is equipped with two independent Bluetooth radios and so a relatively fast communication is possible in the team in a static (i.e. no ad hoc) networking topology. An Area exploration algorithm based on Bluetooth network is also proposed in the paper. In their system, the possible movements of robots are constrained by the change of signal coverage and movements of other subsequent robots.

In [12], authors proposed novel algorithms to navigate robot in an unknown area. The algorithms for processing the laser data give a good result for environment representation and building navigation map in real time. The fusion of laser data with vision and sonar data has good capability to navigate an autonomous search and rescue robot. And [13], based on their previous work, researchers extended their navigation approach using a fuzzy controller that will take paths based on extracted lines and fusing data from sonar modules. The advantage of fuzzy logic is that there are no crisp transition between states, so the system tends to be much more resistant to input

deviations. They also implemented their theory on NAJI V mobile search and rescue robot platform and the results show better performance in localization and mapping as well.

In [14], the authors proposed a navigation algorithm of a robotic system in complex disaster environments. The two-phase motion-planning algorithm is designed for tracked robots with actively controlled actuators to find a fast and stable path to a user-specified goal. For the first phase, an initial path is generated based on terrain roughness. At the second phase, a refined path based on the initial path and robot configurations, such as actuators stability and traction, is generated. Following the final path, the robot can get to the goal position smoothly and quickly.

2.2. Literature Survey on Indoor Search and Rescue System

Current there are several systems [15] that help localize responders, such as firefighters in indoor mass casualty incident and provide responders navigation instructions. In [16], the Fire Information and Rescue Equipment (FIRE) system is developed for urban and industrial firefighting and emergency response. The Telos Sky mote 802.15.4 platform is deployed in a target building and the beacon motes constantly broadcast static information at a frequency of 40 Hz. Based on the pre-calibration data and RSS signals received by "firefighter" mote, the locations of firefighters can be determined. Firefighters can see their location on their head-mounted display.

In [17], a new approach for firefighter navigation support was proposed in the LifeNet system. Instead of deploying beacon based on absolute location on a map, the system did not need a map at all. All the local guidance was based on an ad hoc deployed system, making it flexible and realizable for use in a building without a floor plan.

In [18], the author proposed the FIREGUIDE system using Bluetooth and RFID technology. A zone-based localization was realized to localize firefighters and provide firefighters navigation information to the nearest exits.

In [19], besides building a navigational support tool for firefighters on the basis of a manually deployed ad hoc wireless sensor network, authors also integrated the model view

controller (MVC) design pattern for flexible configuration and continuous object related data storage to evaluate the sequence of events and actions to optimize firefighter's strategy and tactics for future mission.

CHAPTER 3

MOBILE PLATFORM FOR OUTDOOR INFORMATION COLLECTION

In this chapter, I present the DIORAMA mobile platform for outdoor information collection in disaster scenarios. Section 3.1 presents the system architecture. Section 3.2 describes the hardware used to build the mobile platform. Section 3.3 discusses the onboard Android application for controlling the robot. Section 3.4 presents the robot commander application for generating robot control strategy. Test results are shown in Section 3.5.

3.1. System Architecture

In order to realize the automatic outdoor real time information collection, the robot should have the following basic requirements:

- It can be controlled by human rescuer
- It can fulfill its task without a human rescuer
- It can move around in the disaster environment
- It can carry various sensors to collect data
- It can communicate remotely with other device or human rescuers

Based on this, a robot integrated disaster management system is proposed. The overall architecture of the system is shown in Figure.3.

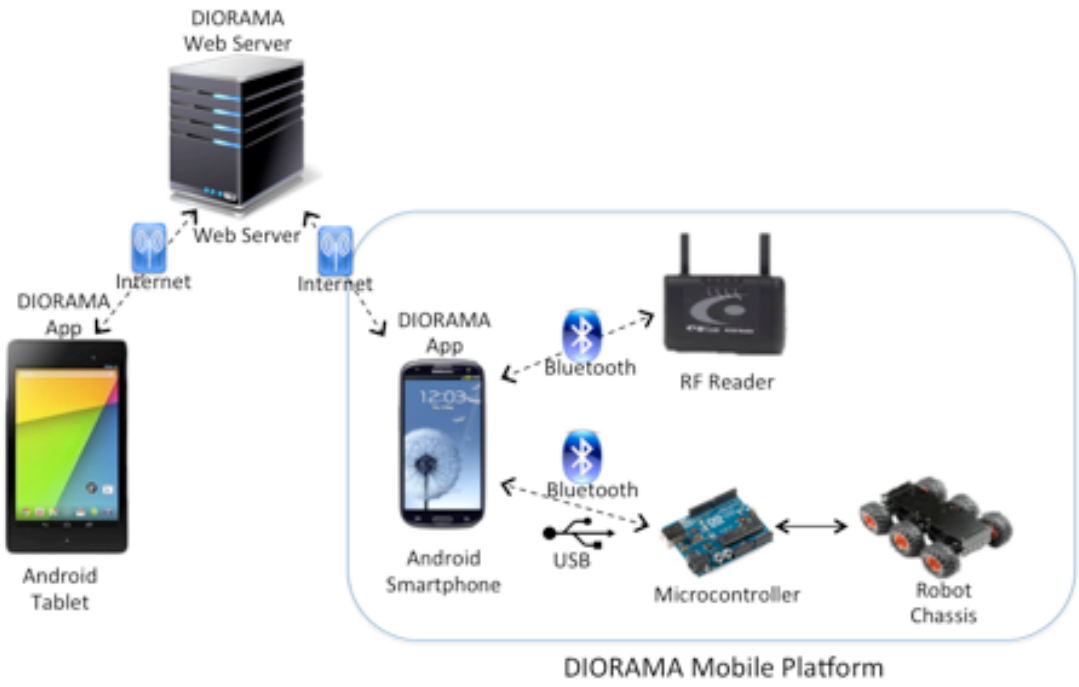


Figure.3. Architecture of Robot Integrated DIORAMA System

As shown in **Figure.3**, the system is consisted of two parts: the DIORAMA mobile platform and the remote control subsystem.

Components of the DIORAMA mobile platform is shown in the blue box, which includes a robot chassis, a microcontroller, an Android phone running the DIORAMA robot monitor application and an active RFID reader. The microcontroller controls the movements of the robot chassis. It connects directly with motors on the robot chassis. Sensors and servos are also connected to the microcontroller. The active RFID reader is mounted on the robot to receive RF readings from the active RFID tags attached to each patient. The Android phone on the mobile platform provides the following functions: 1) communication between the DIORAMA web server and the mobile platform using either 4G or WiFi; 2) to collect all the information necessaryinformation collecction from the RF reader and sensors on the microcontroller; 3) to collect internal sensors' information collection from the phone (e.g. GPS, compass, accelerometer, camera); 4) control strategy generation based on collected data; 5) control the onboard microcontroller which in turn controls the platform movements.

The remote control subsystem contains an Android tablet, the DIORAMA web server and the Android phone on the robot. The DIORAMA robot commander application is installed on the tablet. The tablet and the phone cannot communicate with each other directly. But with the help of the DIORAMA web server, both applications are able to share information and commands with each other via Internet. For example: the location of the mobile platform is obtained from the GPS module on the phone and the orientation is given by the compass also on the phone. The location of the mobile platform is continuously uploaded to server, ensuring that the DIORAMA robot commander application get real time updates on the platform localization.

3.2. Hardware Design

3.2.1. Robot Chassis

A reliable and robust robot chassis is the basis of all the upper level developments. For the purpose of using in disaster management, the robot chassis should have the following basic properties:

- The robot chassis must have motion units, such as motors;
- The robot chassis must have a powerful motion system, which enables the robot to move both in low and high speed situations;
- The robot chassis should be able to move smoothly on all kinds of terrain surface, such as on pavements, on sandy road and on lawn;
- The robot chassis should have high extensibility, which means its structure can be easily modified or be mounted with other sensors and devices;
- The robot chassis should be firm and capable of sustaining some weight;
- The robot chassis is harmless to human beings.

A lot of reliable robot chassis have already been built by researches. In [20], a tank based robot chassis is developed. In [21], the ATRv Junior (PIAP) and the ROBUDEM (RMA) robot are used to assist fire-fighting services. And in [22], a robot chassis called NAJA V is built to test their

path finding and obstacle avoidance algorithm. For our particular DIORAMA integrated robot platform, I choose the already-made Dagu Wild Thumper Mobile Chassis from Dagu Electronics.

As shown in Figure.4, this 6-wheel-drive chassis from Dagu Electronics is designed to excel at traversing rough terrain and steep inclines, making it a great platform for any robot that needs to perform tasks in a disaster environment. It features six powerful DC motors with large spiked tires. A unique “super-twist” suspension system acts to keep each wheel in contact with the ground for maximum traction, even when driving over uneven or bumpy surfaces. The suspension can be adjusted to suit different loads and conditions. The chassis is made from a 2mm-thick corrosion-resistant anodized aluminum plate, all of the nuts, bolts, and screws are stainless steel, and the brass fittings and suspension springs are nickel-plated.

The motors are wired together on a terminal strip inside the 2mm thick anodized metal frame and can be controlled on two channels (left and right). The motors are intended for a maximum nominal operating voltage of 7.2 V (2V minimum), and each has a stall current of 6.6 A and a no-load current of 420 mA at 7.2 V. When powered at 7.2 V, the version of the chassis with 34:1 gearboxes can reach a top speed of approximately 7 km/h (4.5 mph), and each motor has a stall torque of roughly 5 kg-cm (70 oz-in). It is a differential drive platform, meaning that a turn in accomplished by simply driving one side faster or slower than the other, or turning the two sides in opposite directions. The dimension of the chassis is 420 x 300 x 130mm (16.5" x 12" x 5") and the cost is less than 300 USD, which meets our low cost goal.

Figure.4. Dagu Wild Thumper Robot Chassis [23]

3.2.2. Microcontroller

In order to control the robot chassis, I choose Arduino as its onboard microcontroller. Arduino is an open source electrical prototyping platform based on flexible and easy-to-use hardware and software. It is a single board microcontroller and the open source hardware is designed around an 8-bit Atmel AVR microcontroller or a 32-bit Atmel ARM [24].

Since the release of first board in 2005, a wide range of Arduino board has been developed and produced. For my project, Arduino Romeo, which is an Arduino UNO based board, is chosen. The powerful All-in-One Arduino compatible microcontroller is designed for various robot applications. Same with Arduino UNO board, it is based on the ATmega328 and has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It has an operating voltage of 5V with input voltage from 6V to 20V. The flash memory is 32KB, along with 2KB of SRAM and 1KB of EEPROM. In addition to this, the Arduino Romeo board also has an integrated 2-way DC motor driver and a Bluetooth communication interface.

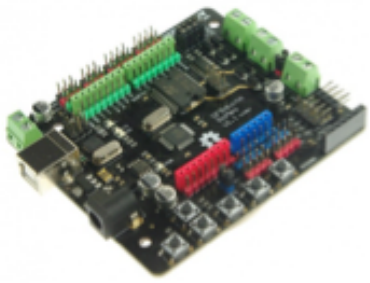


Figure.5. Arduino Romeo Microcontroller

Arduino comes with a simple integrated development environment (IDE) that runs on regular personal computers and allows writing programs for Arduino using C or C++. Also, the Arduino software IDE is user-friendly and simple to use. User only needs to define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings
- loop(): a function called repeatedly until the board powers off

3.2.3. Motor Driver, Communication Module and Sensors

3.2.3.1. Motor Driver

As it is mentioned in Section 3.2.1, the stall current of a single motor is 6.6 A. So when the robot wants to start move from rest, it would require current of as much as 20 A. This amount of current is far beyond what could be supplied by the motor driver on the Arduino Romeo board. An additional motor driver is necessary.

Here the T'Rex robot/motor controller is selected. The T'Rex controller is also from DAGU [25]. It is an Arduino compatible robot controller designed to power and control servos and motors. The ATmega328P controller comes with the Arduino bootloader and customer software can be uploaded via the built-in USB interface, ISP or FTDI sockets. The Dual H bridges are rated for stall currents of 40A per motor and the average current is 18A per motor. Factory calibrated hall-effect sensors measure current draw of each motor. Each motor has independent variable electronic braking. Self-resetting PTC fuses can prevent damage from stalled motors.

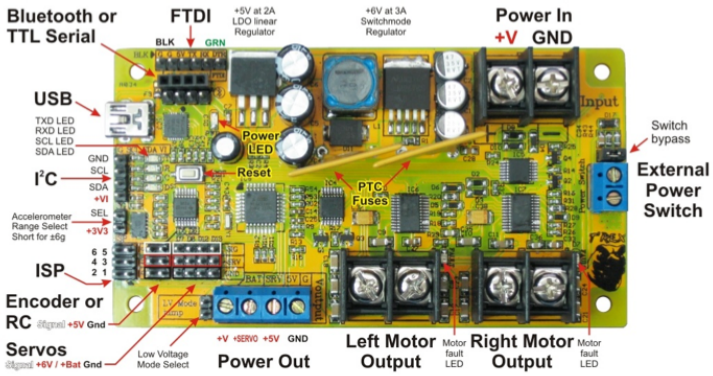


Figure.6. T'Rex Robot/Motor Controller

3.2.3.2. Infrared Sensor

In order to realize the functionality of obstacle avoidance, a range finder is used for the robot to detect whether there is an obstacle on its moving path. There is a lot of choice, such as ultrasonic sensor, Laser range finder, sonar and infrared sensor. For the purpose of simple obstacle detection, the accuracy of the distance between the robot and an obstacle is not so important. Considering this, infrared proximity sensor is good enough. In this project, SHARP long-range infrared proximity sensor is chosen. The sensor has an analog output that varies from 2.8V at 15cm to 0.4V at 150cm with a supply voltage between 4.5V and 5.5V DC. It is good enough to sense objects up to 1.5 meter away. As shown in Figure.8, three infrared sensors are placed in the front of robot chassis. One is heading ahead while the other two are heading slightly to left and right with 135° to the front one respectively. Such setting guarantees the robot is able to detect most of obstacles in its moving path.

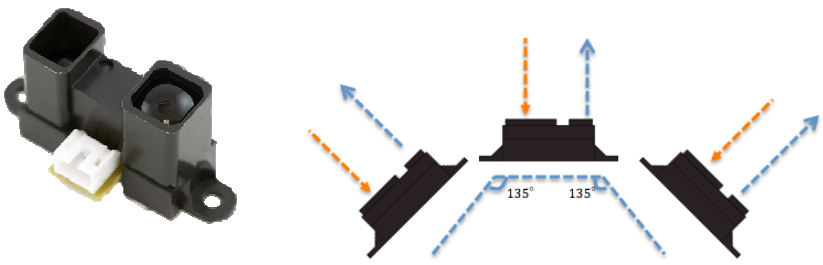


Figure.7. Infrared Sensor

3.2.3.3. Servo

A servo is used to mount Android phone. It can rotate the attached phone holder, enabling the camera on phone get a better view while taking pictures and videos. An easy-to-use servo is shown in Figure.8. It features with 180°rotation degree at the maximum speed of 375° per second.

Figure.8. Servo

3.2.4. Android Operating System

The real "Brain" of the robot is the android smart phone mounted on the robot chassis. The Android device provides sufficient computational capability and memory, which are essential for real time data collection, data processing, path planning, motion control and multi-channel communication. All these tasks are processed within Android operating system on phone.

Android is an operating system based on Linux Kernel. The architecture is shown in Figure.9.

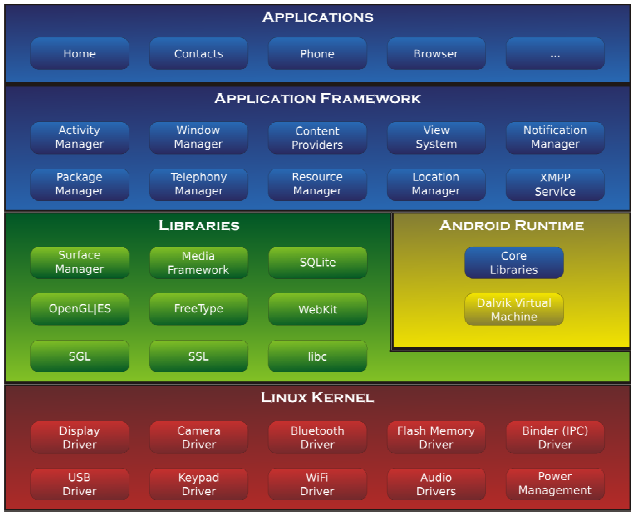


Figure.9. Android Operating System Architecture [from Google image]

The mobile platform as a whole is shown in Figure.10.

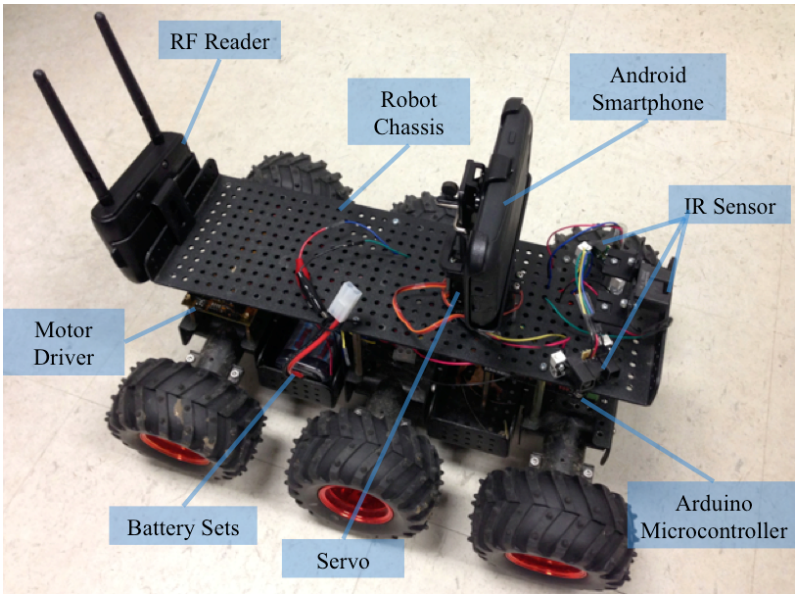


Figure.10. DIORAMA Mobile Platform

3.3. Onboard Robot Application

Although the robot already has a microcontroller, which comes along with limited memory and computational power, it is not enough to fulfill all the tasks. Here I developed a separate application -- the *robot monitor application,* which runs on the onboard Android phone, to control the robot's behavior as well as sense surroundings of robot. At the same time, it also serves as an interface between the robot and the remote control system.

The functionalities of the application is as follows:

- Communicate with microcontroller on robot chassis;
- Send motion control commands to microcontroller;
- Get sensor information from microcontroller;
- Get robot's real time localization information from GPS;
- Get robot's bearing information from compass;
- Collect Radio Frequency readings from active RFID tags on patients;
- Take pictures and capture videos

- Store pictures and videos locally or upload to DIORAMA web server

3.3.1. Communication with Robot

As mentioned above, the primary goal of the application is taking over computational tasks from Arduino microcontroller to an Android smartphone. To achieve this goal, a prerequisite is a reliable data communication connection between Android smartphone and Arduino microcontroller. A reliable data communication connection can guarantee the correctness and integrity of data that collected by robot. It also provides robot a fast and reliable acceptance mechanism. I developed two kinds of communication schemes: one is through Bluetooth and the other is through USB host technology.

3.3.1.1 Bluetooth communication

Bluetooth is a wireless communication standard for exchanging over short distances from fixed or mobile devices. The physical range of Bluetooth communication is as far as 60 meters. As for our implementation, both the Android smartphone and the microcontroller are mounted on the same robot chassis. So the communication quality would not be affect by distance factor.

The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth functionality through the Android Bluetooth APIs. These APIs let applications wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features. Using the Bluetooth API, an Android device can:

- Scan for other Bluetooth devices
- Query the local Bluetooth adapter for paired Bluetooth devices
- Establish RFCOMM channels
- Connect to other devices through service discovery
- Transfer data to and get date from other devices

- Manage multiple connections

When connecting to a Bluetooth module, the digital pin 0 and pin 1 on Arduino will be used as TXD and RXD to send and receive data via Bluetooth. An Arduino program can receive and send data simply through Serial.read() and Serial.print() methods. The Bluetooth module has a unique physical address and can be paired with an Android smartphone.

The communication establishment procedure is shown in Figure.11:

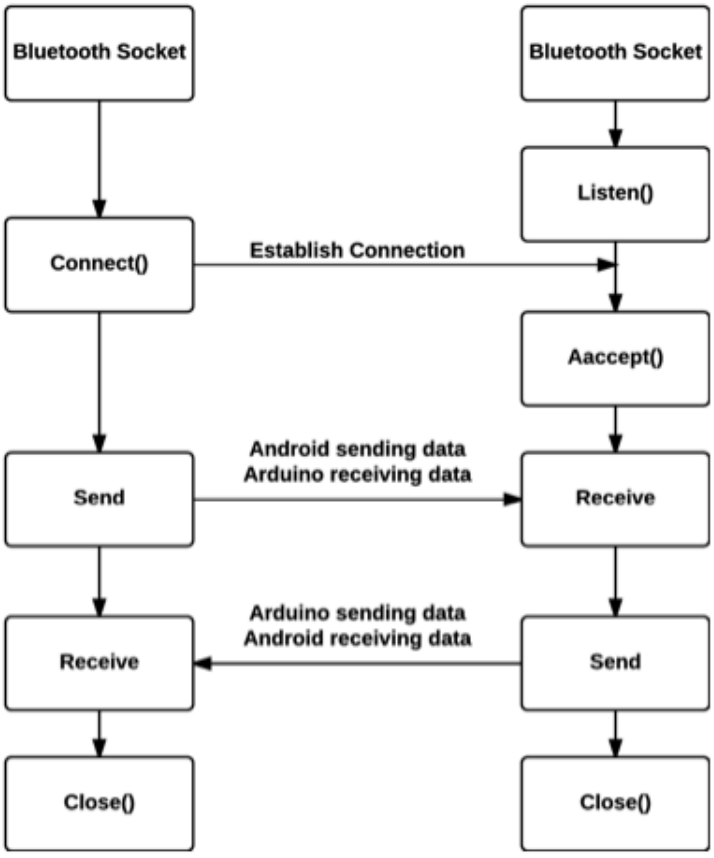


Figure.11. Bluetooth Communication

Before opening the robot monitor application, the Android smartphone should search for a Bluetooth module and then uses valid VIN number to pair with it. As long as the Bluetooth module is visible to the Android smartphone, the phone will try to connect to microcontroller when the application is opened. At this step, the phone is acting as a client who creates an RFCOMM Bluetooth socket with UUID, which is an application-unique ID, and then send communication

request to the Bluetooth module. The module now acts as Bluetooth communication server. Once receiving the request, a RFCOMMM Bluetooth server socket is build and data are read into Arduino's buffer. After communication is successfully built, the monitor application can easily send or receive data using OutputStream or InputStream of Bluetooth socket respectively. It should be mentioned that during the whole process, multiple threads are involved. One thread is to establish the connection, one is used to listen and read data from the Bluetooth socket and the other is to process the incoming data, extracting useful parts and transferring it into readable format.

Besides the communication between Android phone with microcontroller, Bluetooth is also used for communication between RF reader and the phone. The RF reader has already integrated Bluetooth communication module inside, so to retrieve RF readings, the phone only needs to build a separate Bluetooth communication channel using the physical address of RF reader. The process of establishing connection and data retrieving is the same with that of Arduino case.

As it is shown in Figure.13, two Bluetooth services, BluetoothRFIDReaderService and BluetoothRobotService, will be started when the application launches.

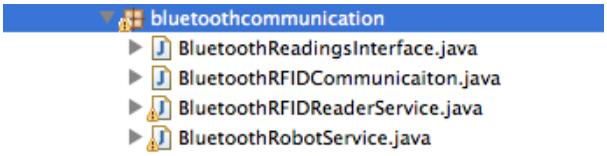


Figure.12. Bluetooth Communication Service for Robot and RFID Reader

3.3.1.2 USB Host

Since not every Android smartphone supports multi-channel Bluetooth communication, here I developed another method to communicate with the microcontroller—using USB Host mode on Android smartphone. USB Host mode enables Android-powered device to act as the USB host to power the bus and enumerates connected USB devices. So when users connect USB devices to an Android-powered device, the Android system can determine whether your

application is interested in the connected device. In order to set up a communication with Arduino microcontroller, the robot monitor application has to:

- Discover connected USB devices by using an intent filter to be notified when the user connects a USB device or by enumerating USB devices that are already connected.
- Ask the user for permission to connect to the USB device, if not already obtained
- Communicate with the USB device by reading and writing data on the appropriate interface

The FTDriver open source library enables Android device to communicate with device that has a FTDI or similar USB chip. So when the Android phone is connected with the Arduino Romeo Board, it automatically starts discovery and then informs user a useful device has been found. But in order to communicate with the microcontroller, appropriate USB interface and endpoint should be specified. As for Arduino Romeo, there are 1 interface and 2 endpoints, one for input and the other for output. After a valid USB connection is opened on an appropriate endpoint, it can use bulkTransfer() or controlTransfer() to send and receive data from the Arduino microcontroller. Similar with Bluetooth communication, multiple threads should be created to send and receive and process data.

3.3.2. Robot Localization

As it is the same with human rescuer, a robot mobile platform should report its real time location to incident commander during the search and rescue process. The location of the mobile platform is obtained from the phone GPS module.

GPS is short for Global Positioning System. It is a spaced-based system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The accuracy of GPS could be as high as several meters, which is good enough for our outdoor search and rescue.

Android OS provides GPS localization service that enable user to get up-to-date location. Within the robot monitor application, a GPS service program has been created and would be started every time the application launches. The robot location is updated every 500 ms. The GPS data returns to application is in the format of Android location object and can be further translated into pairs of latitude and longitude.

The GPS readings are not only to report the robot's current location, but also used to determine whether the robot has reached its destination. In the robot monitor application, a proximity alert is set based on the robot current location and the goal location. When the robot reaches any points about 5 meters to the destination point, the alert will be activated. To get the distance between the robot and the destination from latitude and longitude, Haversine formula is used. The Haversine formula is given as follows:

$$d = 2r * \arcsin\left(\sqrt{sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (1)$$

where d is the distance between two points; r is the radius of the Earth; Φ_1 and Φ_2 are the latitude of point 1 and point 2 while λ_1 and λ_2 are the longitudes of two points.

3.3.3. Robot Bearing Control

Bearing is defined as the angle between the forward direction, and the direction from the object to another object. It typically refers to the direction of some object, as seen by us, compared to our current heading. In our robot platform, the bearing is simply the angle between current robot orientation and the goal destination. Here the magnetic north pole is chosen as the reference point to calculate the bearing.

Most Android smartphone has orientation sensor inside. It provides phone's orientation in the format of azimuth, pitch and roll, as shown in Figure.13. The phone is mounted on the robot as shown in Figure.10, so pitch reading is chosen to determine the robot orientation.

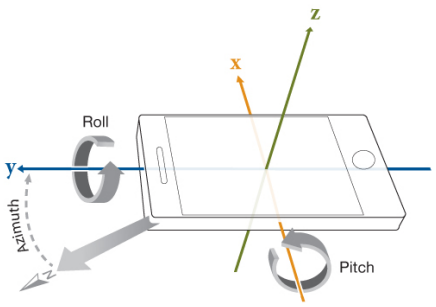


Figure.13. Android Phone Orientation

The bearing is calculated based on the following triangular relationship of three points. In Figure.14, POI denotes the destination while A denotes the robot current location. B is the point where it has the same latitude with A and has the same longitude with POI. So angle B-POI-A represents the goal orientation the robot should be, in order to get to the right destination. As a result, the bearing is the difference between current robot orientation and the goal orientation.

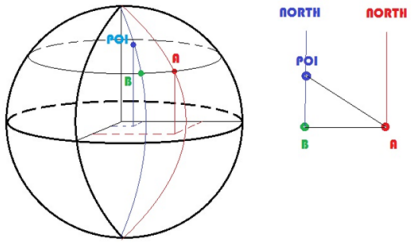


Figure.14. Triangular Relationship Between Robot Location and Destination

By updating the orientation of robot continuously, it is possible to keep the robot moving in the desired direction.

3.3.4. Robot Motion Control

Due to the nature of robot movement, it is impossible to make the robot always move in a straight line without any control mechanism. One-step further, although the robot can make in-place turning on smooth surface, it is easy to get stuck on surface like grass to do so. So a differential turning strategy should be implemented. Two control methods are implemented in this section, one is a simple feedback control and the other is PID control.

3.3.3.1 Feedback control

The theory behind this method is intuitive: when the bearing is within -5 to 5, the robot should move straight forward. If the bearing is less than -5, then the robot should turn left while the bearing is larger than 5, the robot should turn right. When the bearing is -180 or 180, it indicates the robot should make a U-turn from right.

Considering the system error of orientation on phone, a "safe area" is defined as -5 or 5 degrees from goal orientation, as shown in Figure.15. This means when the bearing of robot is less than 5 degrees, it should stop turning and move forward. When the orientation difference is greater than 20 degrees, the robot is regarded deviate from its desired orientation too much and must turn. As for the case when the difference ranges from 5 to 10 degrees, robot should maintain its current motion, whether it is moving forward or turning.

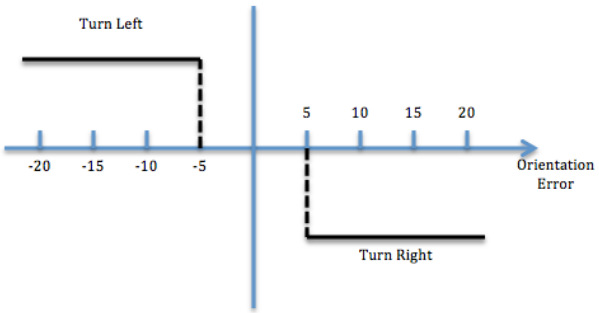


Figure.15. Orientation Feedback Control

3.3.3.2 PID Control

The method introduced in Section 3.3.3.1 is good enough for controlling robot's motion in an open space area. But in a disaster area, there is few open space area. What is more, based on multiple tests in open space, the methods could cause robot spinning around a certain position. To solve this problem, it leads me to the popular PID control strategy.

PID controller is short for proportional-integral-derivative controller. It utilizes feedback control mechanism. An error is calculated as the difference between a measured value and a preset

desired value. The controller takes this error value as input and tries to minimize it by adjusting system input. A block diagram of the PID controller is shown in Figure.16.

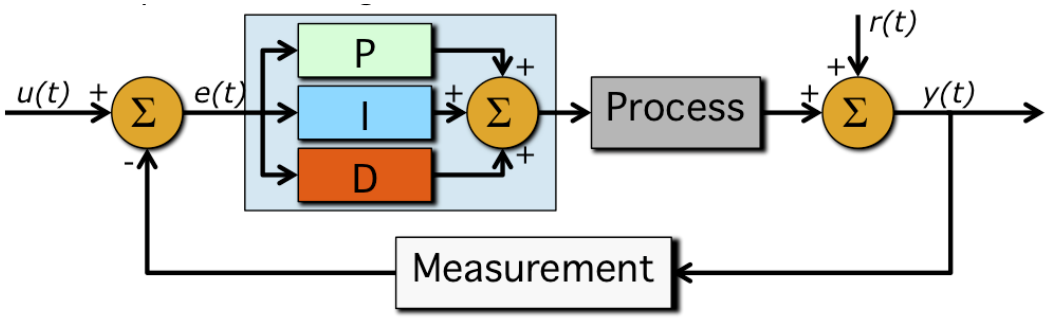


Figure.16. PID Control Flow

As for our robot, $u(t)$ is the system input, which is the desired orientation the robot should be; $y(t)$ is the output of system. $e(t)$ is the error between system output and goal, which is exactly the bearing of robot at time t. Process represents motor motion; $r(t)$ denotes the disturbances. PID controller takes orientation error as input and outputs PWM (Pulse Width Modulation) values that are provided to motors. The three-term controls—proportional, integral and derivative—are in parallel. Control values are summed up together to adjust the input PWM.

3.3.3.2.1. Proportional

The proportional controller depends on current error value. The proportional response can be adjusted by multiplying the error a proportional constant, K_p, as shown in Fomular.(2).

(2)

So the proportional orientation controller can be expressed in the following graph, as in Figure.17.

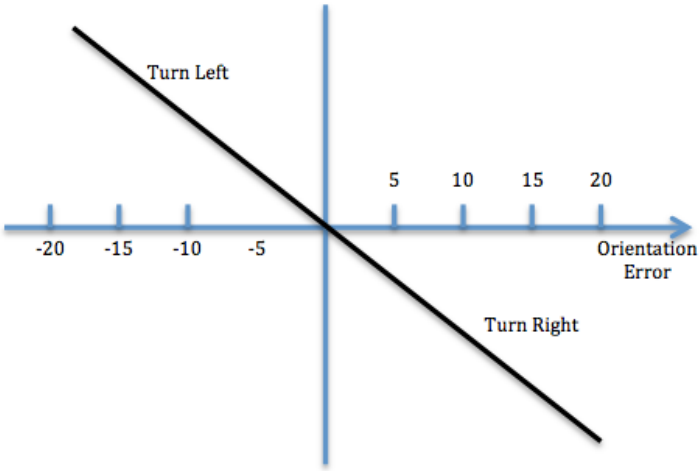


Figure.17. Proportional Control

Proportional controller depends greatly on the value of constant K_p. K_p controls how fast the robot would try to get itself adjust to the right orientation. If it is too small, then the control action may be too small; while if it is too large, the system will become unstable. K_p is a tunable parameter, which means a lot of trials must be done to get the most appropriate value. Here K_p is initially estimated as 0.3, which indicates a single degree the error decreases results in a 0.3 increase in PWM signal.

3.3.3.2.2. Integral

Proportional controller can quickly response to error. But the system only with proportional control always has steady state error, also known as droop. The problem can be solved by adding an integral term to the controller. The integral controller depends on the accumulation of past errors.

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error, as shown in Formula.(3).

(3)

This means the controller generates PWM signal not only based on current orientation error, but takes past errors into account. As a result, it accelerates the controlling process and corrects droops at the same time.

3.3.3.2.3. Derivative

Derivative term is a prediction of future error. The prediction is calculated by determining the slope of the error over time and multiplying this rate of changes by the derivative gain K_d, as shown in Formula.(4).

$$D_{out} = K_d \frac{d}{dt} e(t) \tag{4}$$

The derivative term in the controller can decrease the settling time of the system and improve the system's stability.

So the overall PID controller is as follows:

$$O(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{5}$$

The input to the controller is the orientation error at time t and the output is the value of PWM signal provided by microcontroller to motors.

The value of K_p, K_i and K_d can be tuned based on the "Ziegler-Nichols Methods". It performed by firstly setting K_i and K_p to zero. Then increase the value of K_p to make the system oscillate, meaning that during moving, the robot will turn left and right continuously with similar maximum errors. The value at this point is K_u. Then the oscillation period, T_u, is also measured. K_p, K_i and K_d is set based on the following table:

Control Type	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$1.2K_p/T_u$	-
PD	$0.8K_u$	-	$K_pT_u/8$
Classis PID	$0.60K_u$	$K_p/2T_u$	$K_pT_u/8$

Table.1. Ziegler-Nichols Method

3.3.5. Obstacle Avoidance

It is important for an autonomous robot to detect and avoid collision into an obstacle, especially in the case of search and rescue process.

As mention in Section 3.2.3.2, infrared sensor is used for obstacle detection. The infrared sensor set on the robot can detect obstacle in front with viewing angle of 135 degree. It is sufficient to detect any big obstacle on the forward moving path.

There are a lot of obstacle avoidance method and path finding algorithm. Currently I use a predefined motion pattern to avoid collision with obstacle. Base on the characteristic of the robot chassis and the sensor, the obstacle avoidance motion pattern is defined as follows:

- When an obstacle is detected within 20 cm, the robot moves back off for 1 second.
- Then based on the reading of left and right sensor, the robot turns left or right.

That is, if left infrared sensor reading is larger than that of the right one, robot then turns left. And vice versa.

3.3.6. Immobility Detection

No matter how smart a navigation system and an obstacle avoidance strategy is, it is still possible for the robot to get stuck. Perhaps it bashes into a wall or maybe some appendage got snagged somewhere. Especially during search and rescue in a disaster area, there are rocks or building remains that could easily make the robot stuck. The sensors on the robot may not be able to detect those obstacles. For an autonomous robot, that means game over. There are plenty ways to tell whether a robot gets stuck:

- Wheels are not spinning (Must use an encoder)

It is easy to determine if wheels of a robot are spinning, and how quickly by using wheel encoders. Most of time when the robot is commended to move while the wheels are not spinning, it is safely to say the robot is stuck. But on the other hand, the robot could slip, such as on a sandy surface. The wheels are spinning while the robot never moves, the methods cannot detect and the robot may remain stuck all the time. What is more, there could be a lot of noise in encoder readings.

- Motor current is high

The second method detects stuck by measuring the current on robot motor. When the robot is commanded to move but the wheels of robot are prevented from moving, the current through the motors will increase. A threshold can be set to indicate this situation. But this method also has drawbacks. The current through motor also can be very high when the robot status is changed from static to mobile. At this point the current is equal to the motor stall current. Furthermore, it is hard to determine the actual value of threshold.

- Sensor reading, such as accelerometer

The third method is using the sensor reading to determine whether the robot is stuck. In this case, range sensors can be use. Ideally if the robot is stuck, the reading of the range sensors will remain same, or changes very little. Also with these sensor data, the robot can get a better picture of the environment, which can be helpful when it try to get unstuck. But the disadvantage of this method is also obvious. Sensor reading is subject to noise. And if it is stuck by wall, the range sensors on robot are too close to give correct readings

- Enough time passed

This method is simple but sufficient. A 2 seconds time threshold is set based on the GPS updating rate and robot moving speed. If GPS reading does not change within 2s and accelerometer reading is less than 0.4 (the reading when the robot is static), then the robot is stuck.

How to get unstuck is another issue. Considering the complexity of a disaster area, there might not be any universal solution. Based on current tests in open space, an unstuck moving pattern is designed. As the robot chassis would get its maximum power when it moves forward or backward. So here the stuck handler motion control first make the robot moves back off for 3 seconds, making it possible to get valid GPS reading changes. Then the robot will recalculate its moving path based on the current orientation and position. Up till now this moving pattern works quite well during test.

3.3.7. Pictures and Video Capture

Images and videos of disaster area are helpful not only during search and rescue process, but also in forensic analysis. The robot monitor application integrates functionality of autonomously taking pictures and videos while the robot is moving. More specifically, pictures are taken every 20 meters the robot moves while the video is taken at every destination set by commander. Servo on the robot chassis rotates when taking video and will be reset to its original position after video taking process finishes. The image resolution is 1280×960 formatted in PNG and the video is in 3GP format encoded by H.263 video encoder and AMR audio encoder. The process of taking pictures and videos is in Figure.18. Figure.19 shows a sample image.

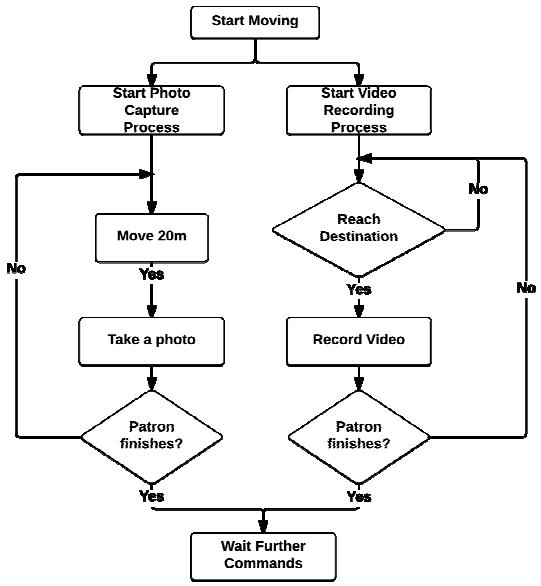


Figure.18. Picture and Video Capture Process

Figure.19. Sample Picture from Robot

3.3.8. Communication with Server

As an interface between remote control system and DIORAMA mobile platform, the robot monitor application is also responsible for communication with robot commander application via web server. The information shared by the Android phone and web server is as shown in Figure.20. The robot monitor application receives the desired destination and the moving path that stored in web server. In return the application uploads robot's real time location as well as locations where pictures and videos are taken. Pictures and videos are also uploaded to server. Multiple threads are created in the application to upload and retrieve these data.

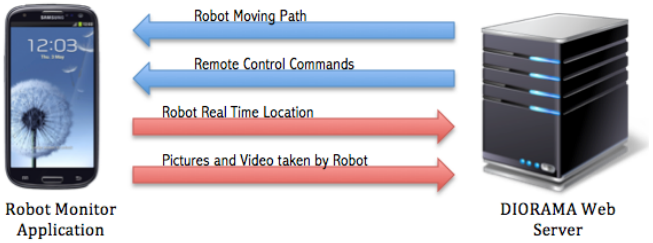


Figure.20. Data Exchange with DIORAMA Web Server

3.4. Robot Commander Application

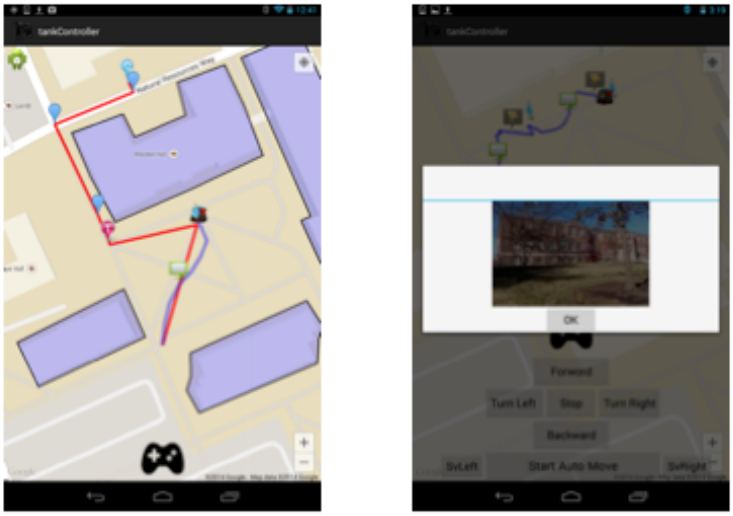
The robot *commander application* is installed on an Android tablet or an Android phone. Both an incident commander and responders can use this application to deploy and control a robot and get information from the robot. This application provides the following features:

- Friendly user interface
- Indication of robot's real time location
- Remote control functionality
- Robot Path Generation
- Picture and video retrieval from server

3.4.1. User Interface

Ultimately, the robot commander system will integrate with the DIORAMA Incident Commander Application. Currently, for the purpose of DIORAMA mobile platform development, an individual robot commander application is designed with its customized user interface.

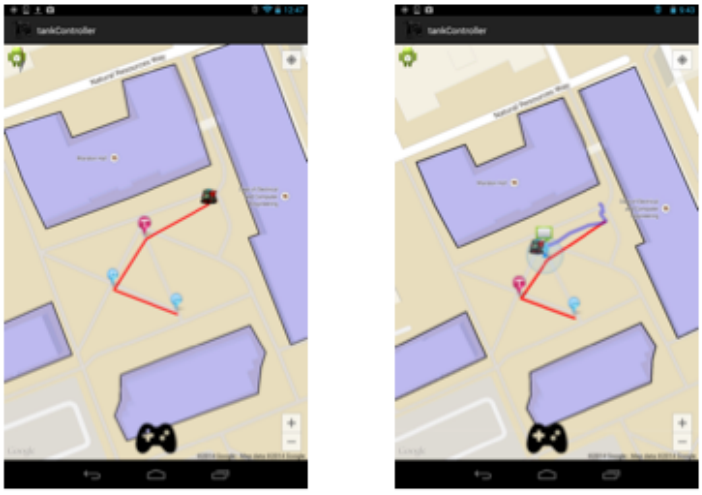
The user interface is shown in Figure.21 to Figure.24. The Google map occupies most space of the screen. It gives user the information of a disaster area as well as real time information of robot. Different markers on map have different meanings. A tank marker indicates the current location of the robot. As the platform moves, the position of the marker changes accordingly. Marker with letter "D" is the platform destination, which is set by the incident commander through a single click on the map. A series of these markers indicate the robot has to reach multiple destinations in a specific order. A marker with letter "T" specifies robot's next destination. This gives commander a better understanding of where the robot is heading. The marker will change to a flag-like marker after the position has already been reached. At the same time, a short video about the robot's current location environment will be able to be retrieved. The incident commander and responders can retrieve these videos by tapping on the video marker.



(a) (b)

Figure.21. Robot Navigation and Image Retrieving

The red polyline indicates the path generated by the commander application. It is the best and the shortest path that the robot should follow. But the map has no representation of rocks or trees, so the real moving path of the robot is different, shown as blue polyline on the map.



(a) (b)

Figure.22. Path Generation and Robot Movement

The purple translucent polygons on map are set manually users, telling the robot those areas are not reachable. This helps a lot for the robot path generation algorithm. To set these polygons, users could simply call the sliding window on the left by first set the boundary of the

polygon and then submit the setting. Polygons can also be modified or deleted. An SQLite database is used to store the coordinates of each polygon.

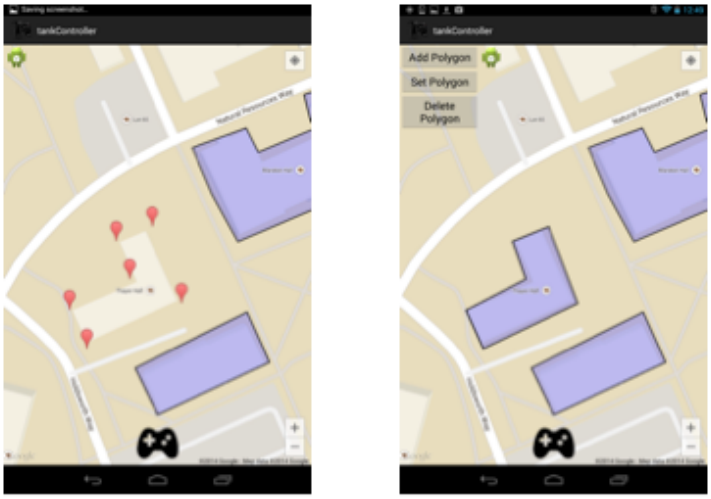


Figure.23. Reachable Area Setting

A control panel is hidden in the sliding drawer at the bottom of screen. Users can set the robot's motion as controlled motion or autonomous motion. Users can also control the servo on the robot chassis. When the robot receives direct motion commands from control panel, it will discard its current preset motion and execute the instant command from users.

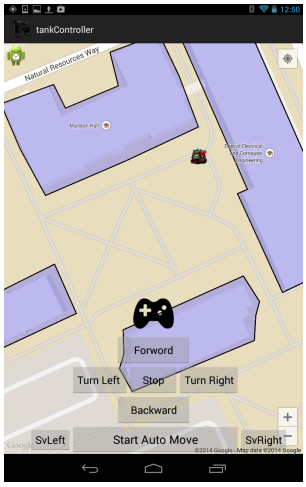


Figure.24. Remote Control Panel

3.4.2. Communication with DIORAMA Web Server

As it is shown in Figure.25, by communicating with the DIORAMA web server, the robot commander application can get the robot's real time location and media information of the disaster area. The planned path and remote control commands are sent from the application to the DIORAMA server.

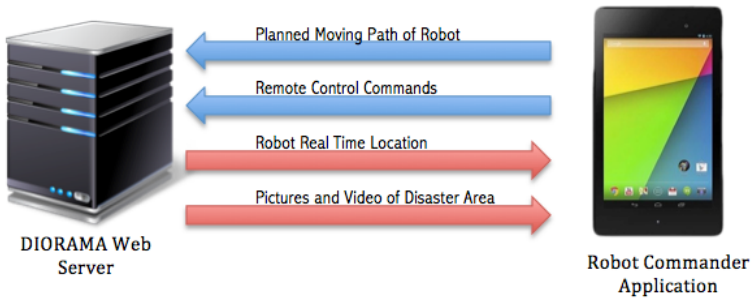


Figure.25. Communication with DIORAMA Web Server

As shown on the UI, the preset moving path of robot is consisted of a series of critical points along with the moving path. They are uploaded to server as a list of coordinate instances. After these information has been sent. A separate thread in the application will be started to continuously monitor the status of these destination points. As soon as the status of a certain point changes, the change will be reflected on the UI. This is the reason why the markers could automatically change its color while the robot is moving. The robot's actual moving path is also obtained by continuously retrieving the robot's locations from server.

Similar with the robot path, the robot commands are also sent via the DIORAMA web server. Currently, the command set is consisted of 5 motion controls, 1 auto/control switch and 2 servo controls.

3.4.3. Robot Path Generation

Path generation is a very important function of the robot commander application. The efficiency of an autonomous robot depends greatly on the robot's moving path. If the path is

poorly planned, the robot could be easily blocked by obstacles. Even with effective obstacle avoidance algorithm, the time for detecting obstacle and calculating detour is not negligible.

3.4.3.1 Path generation V1

The first path generation mechanism depends highly on human rescuer. Since human rescuer have a better sense of the already known obstacles, such as building, he/she can manually set the robot path free from collision to those obstacles. By tapping on map, a series of destination markers will appear on the Google map in the commander application. The robot would follow strictly the order that those destination markers are added, as it is shown in Figure.26.

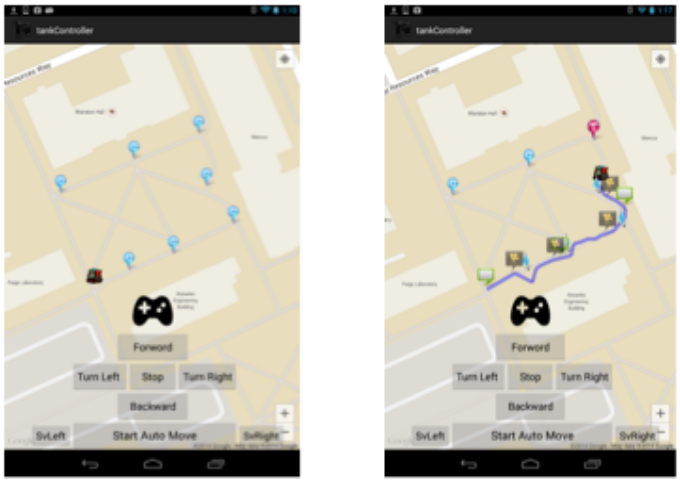


Figure.26. Path Generation Algorithm V1

3.4.3.2 Path Generation V2

But during a disaster search and rescue process, the human rescuer might not have time to set a precise moving path for the robot to avoid large obstacle such as building. So an automatic path generation mechanism is necessary for the robot to quickly get to known which way to take to the target locations or areas.

In order to avoid buildings, it first must know where is the buildings. Unfortunately, Google Map doesn't provide any API to tell the building's location on map. So I come up with a solution -- asking user to select out buildings by drawing translucent polygons on map. Ideally,

this should be done before the search and rescue procedure and information must be stored in database.

Based on the presetting reachable and unreachable area, a path generation algorithm is designed using Google direction API and visibility graph theory.

3.4.3.2.1. Google Direction API

The Google Directions API is a service that calculates directions between locations using an HTTP request. User can search for directions for several modes of transportations, include transit, driving, walking or cycling. In our case, I use walking mode. Directions may specify origins, destinations and waypoints either as text strings or as latitude/longitude coordinates. The Directions API can return multi-part directions using a series of waypoints. The direction is based on existing path on Google map.

As we can see in Figure.27, the path usually starts by finding the nearest point on the surrounding pavement and then follows the pavement to the desired destination.

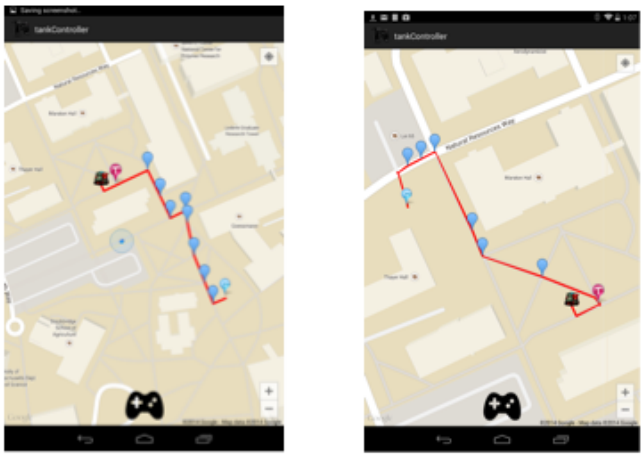


Figure.27. Path Generated by Google Direction API

But a path based on Google map is sometimes not the shortest path. And what is more, as environment changes, Google map may be out of date, making some buildings unavoidable or some paths incorrect. As shown in Figure.28, the robot should have not entered the shaded area,

but the path is planned to directly go across it. This is why visibility graph is also integrated as a complement.

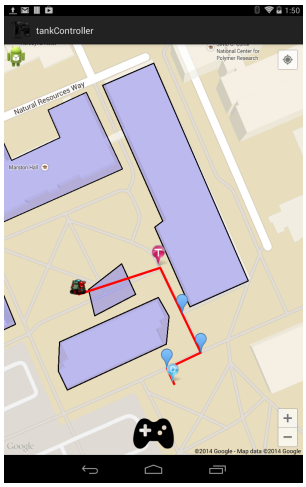


Figure.28. An Error Path Generated by Google Direction API

3.4.3.2.2. Visibility Graph

A visibility graph is a graph of inter-visible locations, typically for a set of points and obstacles in the Euclidean plane. Each node in the graph represents a point location and each edge represents a visible connection between them. In robot motion planning, visibility graph is used to find Euclidean shortest path among a set of polygons. The shortest path between two obstacles follows straight line segments except at the vertices of the obstacles, where it may turn, so the Euclidean shortest path is the shortest path in a visibility graph that has as its nodes the start and destination points and the vertices of the obstacles [26]. Figure.29 and Figure.30 show an example of finding a path within a set of obstacles.

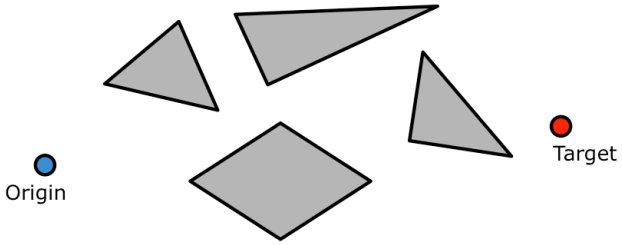


Figure.29. Set of Obstacles between Origin and Destination

There are two steps to find the shortest path based on a visibility graph: first construct a visibility graph and then use a short path algorithm, such as the Dijkstra shortest path algorithm, as shown in Figure.30.

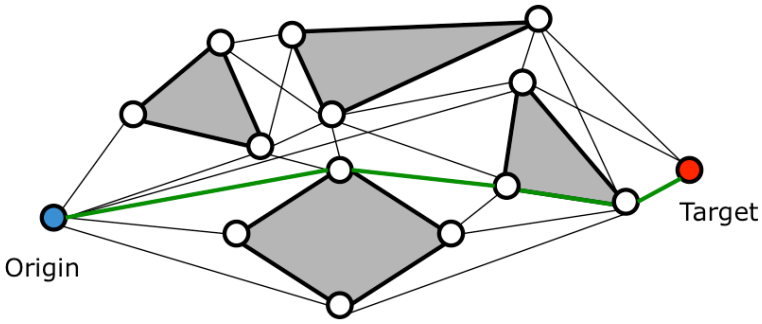


Figure.30. Find Path with Visibility Graph

The algorithm for building a visibility graph is shown in Table.2 to Table.4.

Algorithm VISIBILITY_GRAPH (Plist, S, pstart, pdest)

Input A list of Polygon, A set S of disjoint polygonal obstacles, start point location and end point location

Output The visibility graph Gvis (S).

1. Initialize a graph G = (V, E) where V is the set of all vertices of the polygons in S plus two point and E = empty set.
2. for all vertices v V
3. do W ← VISIBLE_VERTICES (Plist ,v,S)
4. For every vertex w W, add the arc (v, w) to E .
5. return G

Table.2. Algorithm: Building Visibility Graph

Algorithm VISIBLE_VERTICES (Plist, p, S)

Input A list of Polygon, A set S of polygonal obstacles and a point p that does not lie in the interior of any obstacle.

Output The set of all obstacle vertices visible from p.

1. W ← empty set
2. For every other vertices wi in graph
3. do if VISIBLE (Plist ,p, wi) then Add wi to W.□
4. return W

Table.3. Visible Vertices Definition

Algorithm VISIBLE (Plist, p, w_i,)

Input A list of Polygon, two points that are to be calculated

Output A Boolean indicate whether the two points are visible to each other

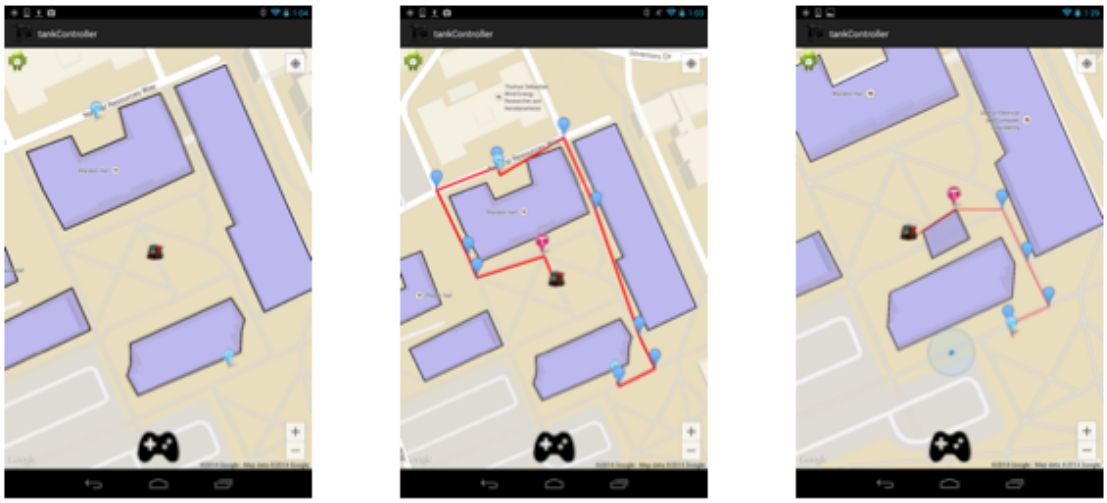
1. calculate distance between p and wi: dis=HaversineDistance (p,wi)
2. if dis<2
3. return True
4. get middle point, p_{mid}, of p and wi
5. for every polygon pg in pList:
6. if p_{mid} is in polygon
7. return false
8. else do VISIBLE (pList, p,p_{mid}) && VISIBLE (pList, wi, p_{mid})

Table.4. Algorithm: Checking Visibility between Two Vertices

So the overall path finding algorithm follows the schema below:

- User first set a series of destinations that the robot should reach in order
- For every pair of destinations, check whether the two points are visible to each other
- If they are, draw path connects the two points directly; if not, use Google Direction API to calculate an alternate path and use visibility graph to check and modify the incorrect path.
- Draw the final path and send the waypoints along with the path to robot monitor application.

With the path generation algorithm, robot can avoid buildings effectively. As shown in **Figure.31(a)**, the robot is in the central of an open-space area. Then human rescuer want the robot first to go to the north of Marston Hall and then across open-space area to go to the south of another building. The preset two destinations are shown as two "D" markers. Commander then triggers the autonomous moving of robot. At this point, a valid path is calculated and then transferred to robot, as in Figure.31(b). Figure.31(c) shows how the robot circumvents the forbidden area.



(a) (b) (c)

Figure.31. Path Generation Algorithm V2

3.4.4. Video and Image Retrieval

Images and videos are captured by robot monitor application and then transferred to DIORAMA web server. By tapping on an image marker or a video marker, corresponding image or video will be download from server and shown in a separate dialog.

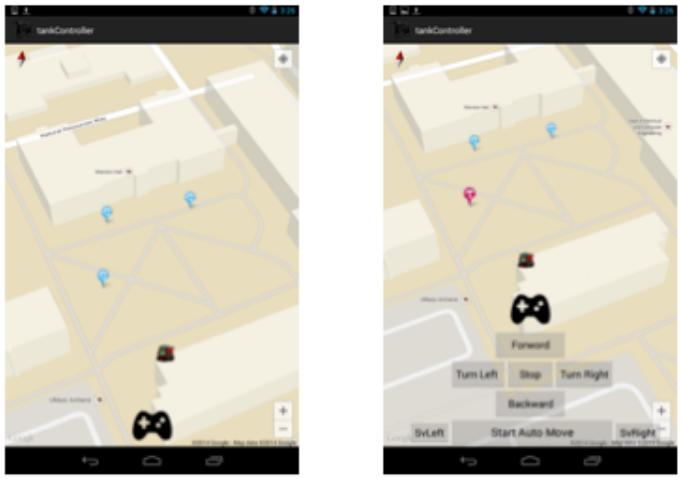
3.5. Testing

Multiple tests have been designed for the DIORAMA mobile platform. The test site is chosen as the open area surrounded by 4 buildings as well as area around the buildings. As shown in Figure.35 the area includes both grass and pavement. There are also benches and streetlights,

which could possibly block the robot's movement. The ground surface is not flat; it has ridges between the pavement and the grass, which may challenge the robot smooth movement.

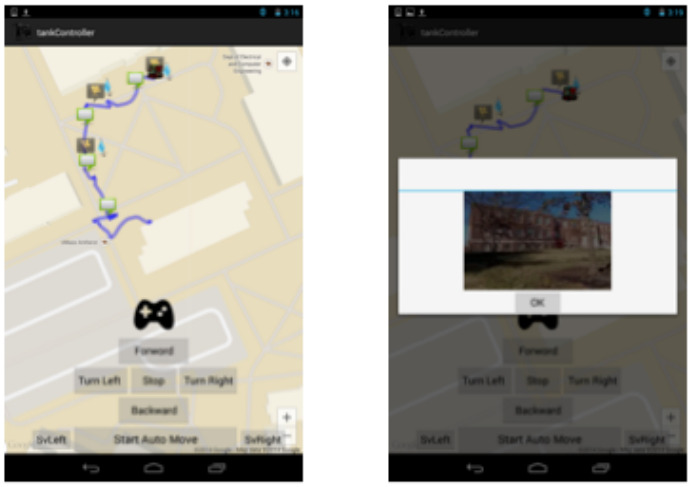
Figure.32. Test Bed

Figure.33 and Figure.34 show the process of an intact test. In Figure.33(a), a human rescuer first set up the path of the mobile platform by providing 3 destinations. Then in Figure.33(b), the command of starting patrolling is sent to robot and the first destination marker changed to red as its current target destination. Figure.34(a) shows the scenario after the robot reached its last destination. All the destinations changed to blue flag and several picture markers and video markers are placed on the map. As Figure.34(b) shows, it is easy to retrieve video or picture through the user interface. Figure.35 and Figure.36 show several other testing results.



(a) (b)

Figure.33. Test Scenario One: Presetting



(a) (b)

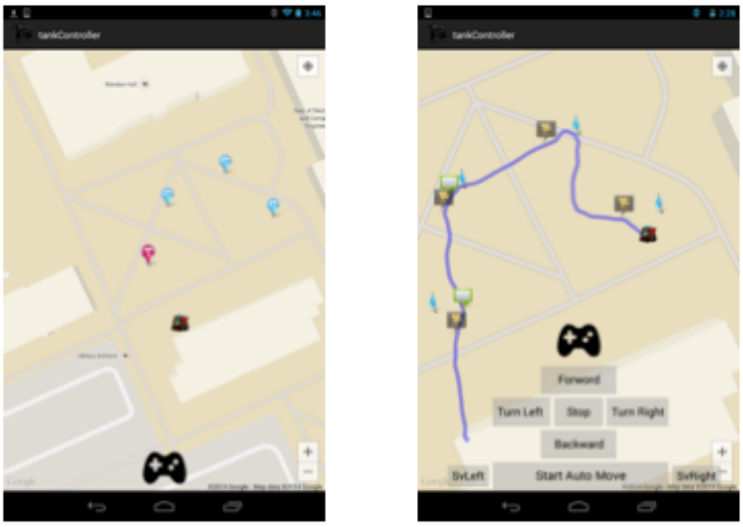
Figure.34. Test Scenario One: Task Finished

After multiple successful tests it can be concluded that the robot:

- moves smoothly on both pavement and grass
- receives and follows commands from the commander application, e.g. the robot moves to desired destinations determined by the incident commander or responder reports current location to the server
- gathers RFID readings while moving and communicate them to the server
- takes pictures and videos and sends them to the server
- The commander application can successfully interact with the robot, can control the robot's movement, set destinations and retrieve the pictures and videos.

(a) (b)

Figure.35. Test Scenario Two



(c) (d)

Figure.36. Test Scenario Three

CHAPTER 4

INDOOR LOCALIZATION AND NAVIGATION OF EMERGENCY RESPONDERS

In this chapter I present a system for quickly localizing responders in an indoor environment and providing navigation instructions. Section 4.1 describes the BLE and the overall structure for the indoor localization system. Section 4.2 introduces the RSSI propagation model. Section 4.3 talks about the indoor beacon deployment strategy. Section 4.4 presents the indoor localization algorithms. Section 4.5 describes the navigation system.

4.1 BLE and System Structure

In this section, Bluetooth Low Energy (BLE) technology and its potentiality to be used in an indoor search and rescue scenario is described. One of the most popular BLE beacons, Estimote is also investigated. Its specification and the reason I choose this beacon for the indoor localization are discussed. The overall system structure is also presented.

4.1.1. Bluetooth Low Energy

Bluetooth Low Energy, also known as Bluetooth Smart, is a wireless personal area network technology introduced in 2011 along with Bluetooth 4.0. As the same as classic Bluetooth, BLE also use 2.4 GHz radio frequencies for communication and data transmission. By using adaptive frequency hopping, BLE device can detect existing signals in ISM band, such as Wi-Fi or other wireless technology using the same spectrum, and avoid them. But compared with classic Bluetooth, BLE has extremely low power consumption, which enables a BLE device to run on a small battery for years. While classis Bluetooth is used mainly for providing a robust wireless connection between devices, BLE has a much less data transfer throughput. Some typical applications for BLE include health care profile, sports and fitness profile as well as proximity sensing.

One of the commercialized BLE technologies is iBeacon, which is a trademark for an indoor position system introduced by Apple Inc. Most of current existing BLE beacons implement the iBeacon standard, which is compatible for both iOS and Android devices.

Typically, a BLE beacon uses proximity sensing to transmit its universally unique identifier (UUID), combining with a major and a minor ID, as shown in Table.5. It will be picked up by a compatible application or operating system. Then the IDs can be then looked up in the device or over the Internet to determine the proximity physical location of the receiving device.

Field	Size	Description
UUID	16 bytes	Application developers should define a UUID specific to their app and deployment use case.
Major	2 bytes	Further specifies a specific iBeacon and use case.
Minor	2 bytes	Allows further subdivision of region or use case.

Table.5. iBeacon Identifier

4.1.2. Estimote Beacon

Currently there are a lot of types of BLE beacons. When choosing BLE beacons, several criteria should be taken into consideration:

- Battery life
- Broadcasting range
- Compatibility with Android
- Configurability
- Size
- Cost

Considering all the above aspect, the Estimote beacon is chosen for the proposed indoor position system.

Figure.37. Estimote Beacon

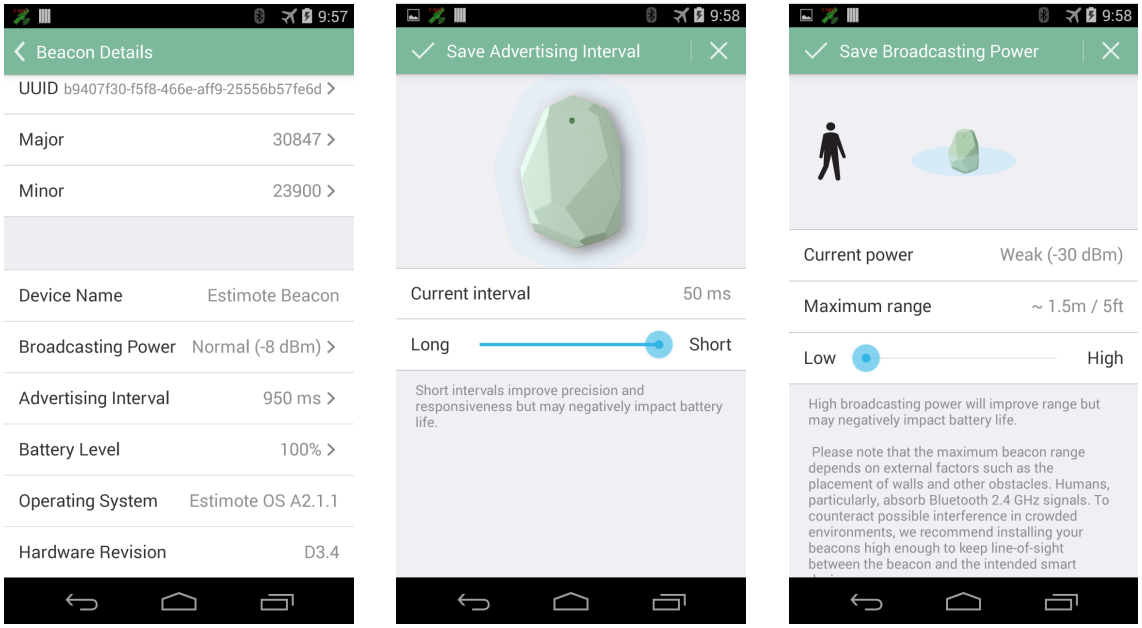
Some main specifications of the Estimote beacon are shown in Table.6:

Identification (model number etc.,)	Estimote model REV.D3.3 Radio Beacon.
Frequency range	2400 MHz to 2483.5 MHz
No. of preset switchable channels	40
No. of voice/data/TV channels	40 Data channels (including 3 advertising channels)
Tx-Rx channel separation	2 MHz
Adjacent channel separation	2 MHz
Frequency stability	<20ppm
2nd Harmonic radiation's	<25 dBuV
Mode of emission	no more than 20 DB
Bandwidth of emission	500 KHz
Type of modulation to be required	GFSK
Power output	4 dBm
Sensitivity	-93 dBm

Table.6. Estimote Beacon Specification

The default battery life for the Estimote beacon is 3 years and the maximum is more than 5 years. Battery is easy to change afterwards. Depending on the broadcasting power, the broadcasting range varies from 1 meter to 70 meters. With its iOS library and Android SDK, it is easy for developer to develop applications using Estimote beacons and smartphones.

One big advantage of Estimote is its configurability. With the Estimote app, the broadcasting power and advertising interval can be configured for different use cases. The UUID, major ID and minor ID are also configurable. Figure.38 shows the Android version of Estimote configuration app.



(a) Identifier Config **(b) Advertising Config** **(c) Broadcasting Config**

Figure.38. Estimote Configuration App

With the size of less than 2 inches, the beacon can be attached to any location or object. Currently the develop kit sold by Estimote is $99.00, comes along with 3 beacons.

4.1.3. Structure of the Indoor Localization and Navigation System

The structure of the proposed indoor localization and navigation system is light-weighted. Based on the size of the localization area, multiple Estimote beacons are deployed. An Android phone is used as a RSSI receiver to collect data and predict a responder's location. It

should be noted that all the computations of the localization algorithms are implemented locally on the smartphone. No server-client based architecture is needed for the localization and navigation. Except several BLE beacons and an Android smartphone, no extra device or sensor is needed in our system. This makes the system light-weighted and ease to use. It is easy to scale to any kinds of indoor environment and the deployment and maintenance cost is also low.

4.2 RSSI and Propagation Model

In the context of range based indoor localization system, the first step is to measure the distance between the unknown node and some fixed references. Usually, these references are made up of beacons or wireless access points. The distances can be estimated based on received signal strength indication (RSSI), which is one of the most popular wireless signal parameters.

4.2.1. Received Signal Strength Indication

In telecommunications, RSSI is a measurement of the power level being received by antenna. The value for RSSI is a signed 8-bit integer varies between -127 to +20, where an increasing value indicates a closer distance. It is usually measured with power level in dBm, which is an abbreviation of power ratio in decibels of the measured power referenced to one milliwatt.

In an Android device, it is very easy to retrieve RSSI values from a BLE beacon. After initializing a BLE scanning service, the device will automatically scan and discover all the nearby beacons. The scanning interval can be set from 50ms to arbitrarily long. This is usually set according to the beacon's advertising interval. As described in the previous section, the adverting interval of Estimote beacon is set to 950ms. Therefore, scanning interval of Android device is set to 1000ms, which is the default value.

4.2.2. RSSI Propagation Model

In order to get a decent indoor localization result, a good propagation model that converts RSSI to distance is a prerequisite. A lot of methods have been proposed to build a good propagation model [27, 28]. Usually, a RSSI model focuses on predicting the average RSSI at a

given distance from the transmitter, as well as the variability of the signal strength in close spatial proximity to a particular location [29]. The log-distance path loss model is one of the most used radio propagation model inside a building or a densely populated area. The log-distance path loss model is formally expressed as:

$$PL = P_{Tx} - P_{Rx} = PL_0 + 10\gamma log_{10}\frac{d}{d_0} + \chi_\sigma \quad (6)$$

where

$\boldsymbol{PL}$ is the total path loss measured

$\boldsymbol{P_{Tx}}$ is the transmitted power

$\boldsymbol{P_{Rx}}$ is the received power

$\boldsymbol{PL_0}$is the path loss at the reference distance $\boldsymbol{d_0}$

$\boldsymbol{d}$ is the distance between transmitter and receiver

$\boldsymbol{\gamma}$ is the path loss exponent

$\boldsymbol{\chi_\sigma}$ is a random variable following Gaussian distribution with zero mean.

As the mean of $\boldsymbol{\chi_\sigma}$ is 0, so it can be eliminated by averaging the RSSI samples. And usually $\boldsymbol{d_0}$ is chosen as 1 meter or 1 foot. So the above Formula transforms into:

$$RSSI = -(10nlog_{10}d + A) \quad (7)$$

where

$\boldsymbol{RSSI}$ is the received RSSI reading on Android device

$\boldsymbol{A}$ is the average RSSI measured at reference point

So the distance from transmitter and receiver can be calculated as

$$d = 10^{\frac{A+RSSI}{-10*n}} \quad (8)$$

In order to determine the value of *A* and *n*, a calibration process is taken. Basically, several beacons are attached to the wall in the target indoor environment at breast level. Then 7 points are chosen to collect RSSI data. There is 3 feet between every two points and 1000 data is collected at each point.

Using linear regression, we can get the following curve in Figure.39.

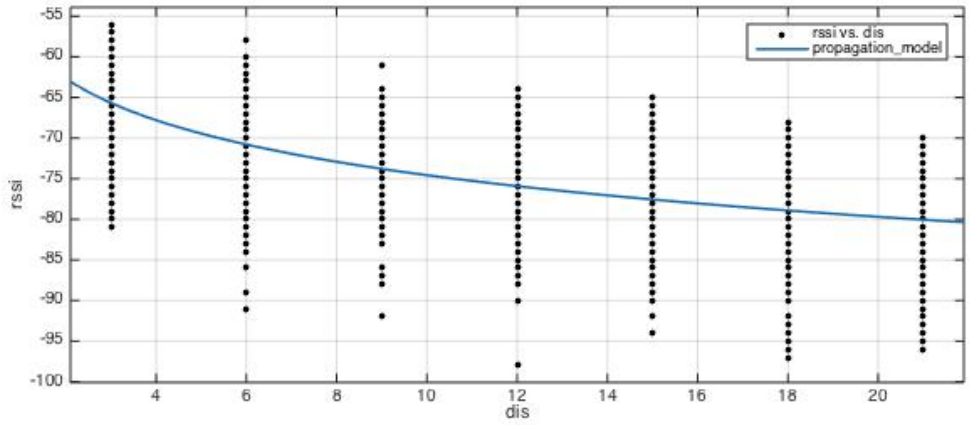


Figure.39. Curve Fitting Based on RSSI Data

So the propagation model is expressed as

$$RSSI = -(10 * 1.702 * log_{10}d + 57.55) \quad (9)$$

And the distance in foot can be calculated from RSSI as

$$d = 10^{\frac{57.55+RSSI}{-10*1.702}} \quad (10)$$

4.2.3. Model Analysis and Improvement

From Figure.39, we can see the RSSI signal is not stable. At each fixed calibration point, the reading can vary more than 20 dBm. Some statistical metrics of the data set are shown in Table.7.

Distance (feet)	3	6	9	12	15	18	21
Mean	-66.300	-70.907	-73.147	-75.227	-76.256	-78.990	-81.887
Variance	29.032	32.998	27.685	25.053	43.128	33.262	41.590
Median	-66	-71	-73	-75	-76	-79	-82
Max	-56	-58	-61	-64	-65	-68	-70
Min	-81	-91	-92	-98	-94	-97	-99

Table.7. RSSI Readings at Different Distance

The noise during RSSI collection has a high impact on the performance of the propagation model. This noise comes from difference sources. Measurement accuracy of RSSI of the most today's radio chips used in sensor nodes is about ±4 dBm [30] and RSSI is also rounded to integer values. And for different environments, the RSSI readings change due to difference structures as well as different construction materials. The noise and uncertainty also come from the instability of the BLE beacons. An instant signal strength burst can cause reading increases irregularly. Furthermore, human body is also a main factor to signal attenuation.

Considering all of these factors, an RSSI threshold is implemented in the indoor localization application. In another word, a filter is added to filter out unreliable readings. As the distance between transmitter and receiver is shorter, the noise of the measurement is smaller [28], so some small RSSI values would be filtered out. The result is shown in Figure.40.

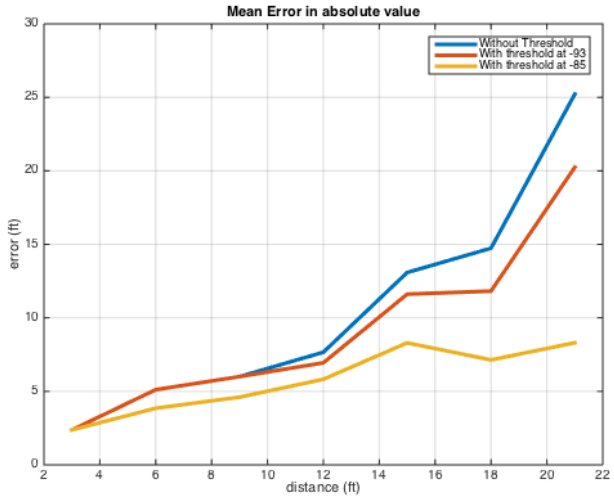


Figure.40. RSSI Errors with Different Thresholds

The error in Figure.40 is calculated on the testing set. Here we can see when a threshold is added, the distance error will decrease. Although a threshold set at -85 looks much better, during the real scenario such threshold will make the receiver neglect a lot of data. As a result, much more time is required to collect enough valid data for localization. As a tradeoff, -93 is chosen as the threshold. All the following deployment strategy, localization algorithms and testing result are based on this threshold.

4.3 Beacon Deployment

As the system uses BLE beacons to broadcast RSSIs and help localization, the priority is to find out where to deploy all the beacons. Similar to Wi-Fi access point deployment, there are two most importance factors to notice:

- Every place in the localization area should be cover by at least one beacon;
- Use the least number of beacons.

That is to say, the beacon deployment should guarantee the localization performance at the minimum hardware cost.

Also, considering the deployment time, beacon deployment can be categorized into pre-deployment and fast-deployment. In the pre-deployment, all the beacons are deployed prior to an indoor mass casualty incident. The positions of beacons are well planed and fixed according to a specified indoor environment. While in the fast-deployment scenario, beacons are deployed on-demand. This is extremely useful during a mass casualty incident. For example, in an accident such as fire, pre-deployment beacon might be damaged or malfunction during the rescue process, so a quick replacement or a re-deployment is necessary. Also in places where no pre-deployment is available, a fast-deployment can quickly adapt itself to the new environment and provide indoor localizations for rescuers.

4.3.1. Deployment Density Analysis

In this thesis, both pre-deployment and fast-deployment are studied for a corridor-like environment. Unlike large open spaces, the structure of the corridor is more linear, so a traditional hexagonal deployment is not applicable in this case. In order to adapt to the linear nature of this environment, the beacons are deployed throughout the corridor based on a straight line. A fixed distance d is pre-defined between every two beacons. Intuitively, the denser the deployment, the smaller the d is and the more accurate the localization should be. But at the same time, more beacons would be used and the cost increases. The relationship between the number of beacons and the deployment density is shown in Table.8. Using the localization algorithms, which are

going to be discussed in Section 4.4, we get the localization performance for different densities in Figure.41.

Number of beacons	Beacon distance interval (m)
13	4
9	6
7	8
5	10
4	15

Table.8. Beacon Number and Deployment Density Relationship

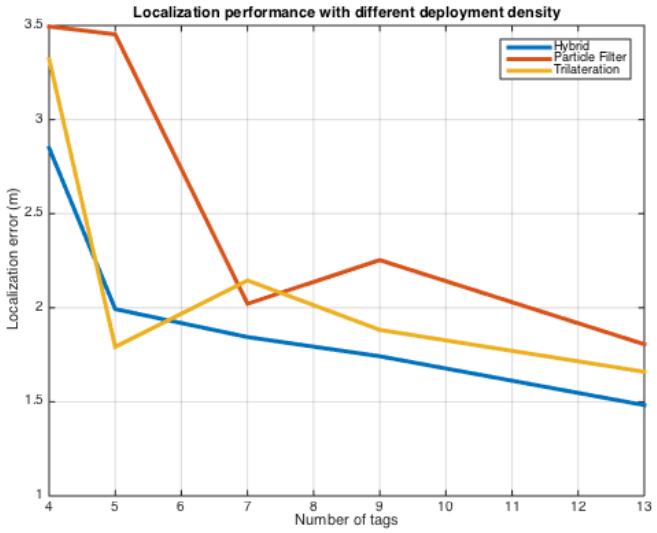


Figure.41. Localization Performance For Different Deployment Density

As we expected, for all of the three localization methods, the more beacons we use, the less error we get from localization. While as the cost for different density is shown in Figure.42, if we want to achieve the least localization error, at least 13 beacons must be used. Taken both localization accuracy and cost into consideration, a deployment strategy of 7 beacons in total is found to be the most cost-effective plan. Such deployment has a slightly higher error while uses nearly half of the beacons compared to the densest deployment.

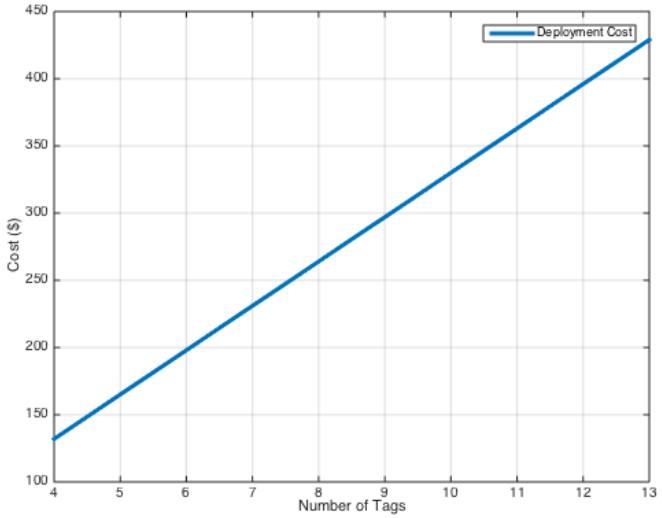
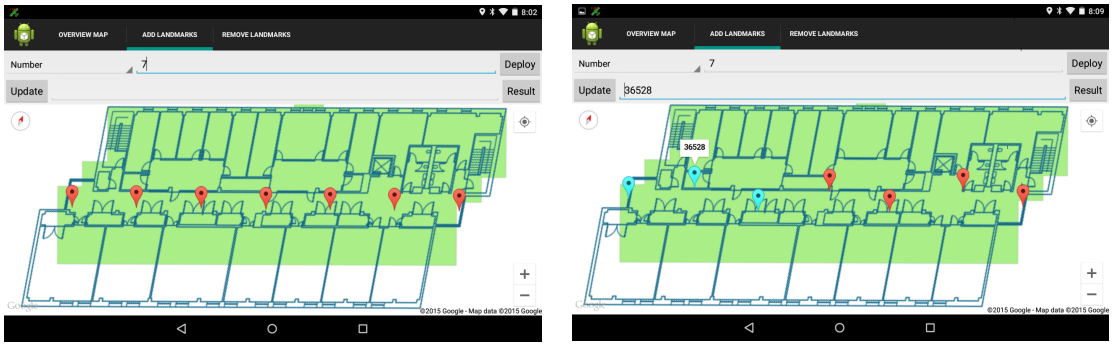


Figure.42. Cost For Different Deployment Density

4.3.2. Fast-deployment Application

Instead of cost-effective factor, time and flexibility is considered more in the fast-deployment scenario. As the same as pre-deployment, each beacon is also assigned a geographical position during fast-deployment. But the distance between every two pair of beacons is not necessarily the same and the position can be changed easily and quickly. An android application is developed to realize fast-deployment. As it is shown in Figure.43(a), as long as the user sets the total number of beacons and two anchor positions of the corridor, the positions of all the other beacons will be generated automatically. The generated positions act as a quick guideline. User can easily adjust the positions of beacons to put them on the wall or some other places by dragging those beacon markers on the map. To add additional tags, user can simply tap on the map. One last essential step is to record minor ID for each deployed beacon in the application. Such ID is essential for the localization algorithms to identify the landmarks and retrieve their positions. As shown in Figure.43(b), marker changes its color to cyan to indicate it has been assigned a minor ID.



(a) Generate Deployment Positions **(b) Adjust Location and Assign Minor**

Figure.43. Beacon Fast-deployment Application

The output of the fast-deployment is in the following format:

@minorID_latitude_longitude_[neighborID1, neighborID2, …, neighborIDn]

The "@" is used to separate each beacon, which followed by the beacon's minor ID. Then latitude and longitude are stored. The last part is a list of minor IDs of the neighbor beacons. The definition of neighbor beacon and its use will be discussed in Section 4.4.

This application adds more agility to the deployment process. Rescuers can quickly get to know where to deploy beacons in a most efficient way. Even if some beacons become out of order or be destroyed, this app makes it convenient to replace those beacons or modify the deployment plan.

4.4 Indoor Localization Techniques

Localization is the core part of the indoor search and rescue application. A good localization result not only indicates the up-to-date position of responder, but can also provide responder a navigation route to quickly find specific rooms and exits.

The use of RSS-based wireless technology, such as Wi-Fi and Bluetooth, for indoor localization has caused a lot of interest among researchers. Different localization algorithms have been developed to get an accurate indoor localization result. In [31], the authors proposed an indoor localization system using Wi-Fi fingerprint technique for mobile device and achieved an average of over 80% localization prediction accuracy. Also, in [32], the author implemented an indoor localization system based on Bluetooth fingerprinting and achieved localization accuracy

that more than 50% predictions were within 1.5m of the actual position when the mobile device did not move.

In [33], the authors described in detail the mythology of using trilateration algorithm to determine the position of users in indoor area based on Wi-Fi signal strength. In [34], the authors investigated and implemented trilateration algorithm. They also discussed about the relationship between the localization accuracy and the number of Wi-Fi access point. In [35], the authors defined a RSS and triangulation based positioning scheme and implemented three distance based triangulation algorithms.

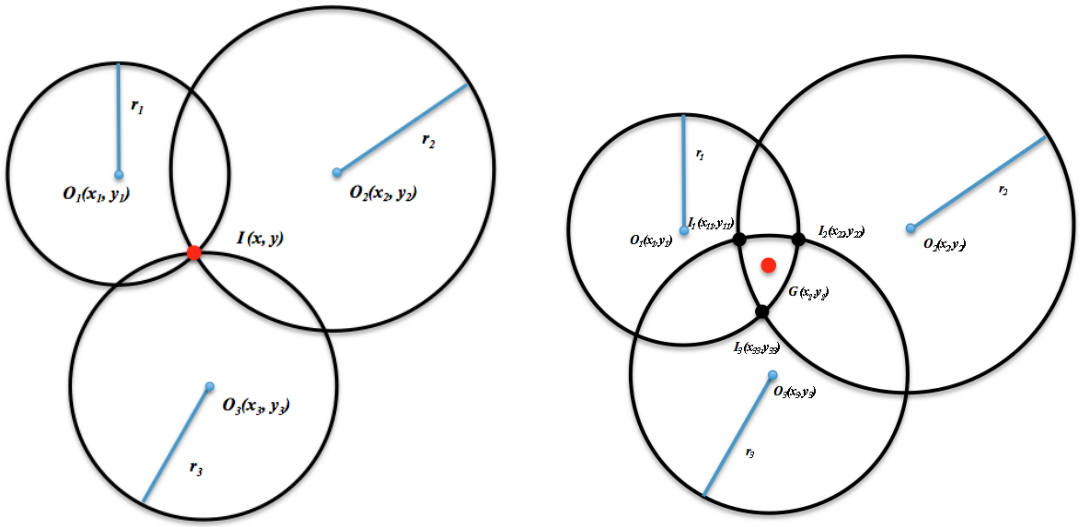
The filter-based technology is also very popular. In [36], the authors proposed using particle filter for simultaneous mobile robot localization and people tracking. In [37], the authors combined trilateration and particle filter. A very accurate result was achieved for localization of mobile robots.

In our work, trilateration and particle filter are chosen to generate localization results based on the RSSI reading from BLE beacons. A hybrid approach combining of the two methods is also proposed in this section.

4.4.1. Trilateration

Trilateration is one of the most renowned and widely used methods for localization and navigation. In trilateration, distance metrics should be measured from the unknown node to three preset BLE beacons. In order to decrease the bias introduced by other beacons, three beacons with highest RSSI readings are used. The position of the unknown node is therefore determined by the intersections of three circles. The centers are the position of the BLE beacons and the radii are from the distance metrics.

With perfect information, the method will give out an exact unique answer. As shown in Figure.44(a), the three circles intersect at exactly one point *I(x, y)*. Therefore, the position of *I* can be calculated from Formula.(11) [38].



(a) Unique Intersection **(b) Multiple Intersections**

Figure.44. Trilateration

(11)

With imperfect information, the circles will intersect at more than one point. As it is shown in Figure.44(b), the position of the unknown node is predicted as the gravity , or centroid of the triangle, which is formed based on the three intersection points I_1, I_2 and I_3.

(12)

But it could also be the case that none of the circles intersect with each other. In [33], it uses Least Square Estimation to look for the position that minimizes the sum of distances to all circles. While in this thesis, another approach is developed to generate the predicted position of the unknown node. As the same as the previous approach, three beacons with highest RSSI readings are chosen. Instead of making three circles, only one circle is made, centering at the position of the beacon with highest RSSI reading. Then 360 points are chosen from the circle, representing possible positions for the unknown node. For each of these points, the sum of distances between the other two beacons (b2 and b3) is calculated. On the other hand, the measured distances to b2 and b3 can be transformed from RSSI readings. So at each point, a

weight is set as the inverse of difference between the calculated distance and the measured distance. Therefore, the position of the unknown node is predicted at the point that has the highest weight.

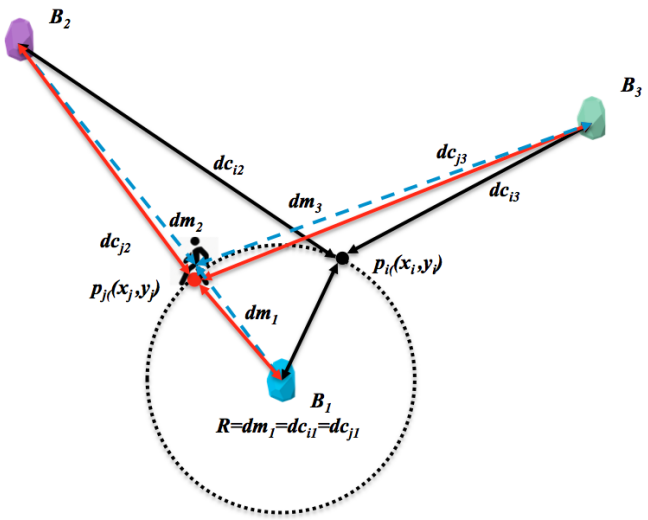


Figure.45. Modified Trilateration Algorithm

As it is shown in Figure.45, B_1 is the closest beacon to the unknown node. So a circle is drawn based on the position of B_1 and dm_1. Comparing two points p_i and p_j on the circle, it is clear that

(13)

where dm_k is the measurement distance to kth beacon and dc_{ik} is the calculated distance between the ith point on the circle and the kth beacon. Since the weights of p_i and p_j are defined as

(14)

therefore $w_j > w_i$ and point p_j is more likely to be the position of the unknown node.

In this approach, the localization of the unknown node relies more on the closest beacon. In another word, the closest beacon acts as an anchor. The other two beacons are used to determine the predicted position from a set of calculated points. This is reasonable since the closer to a BLE beacon, the more reliable the RSSI received from that beacon. At the same time, less bias is introduced from the other two beacons.

The overall flowchart of this approach is shown in Figure.46.

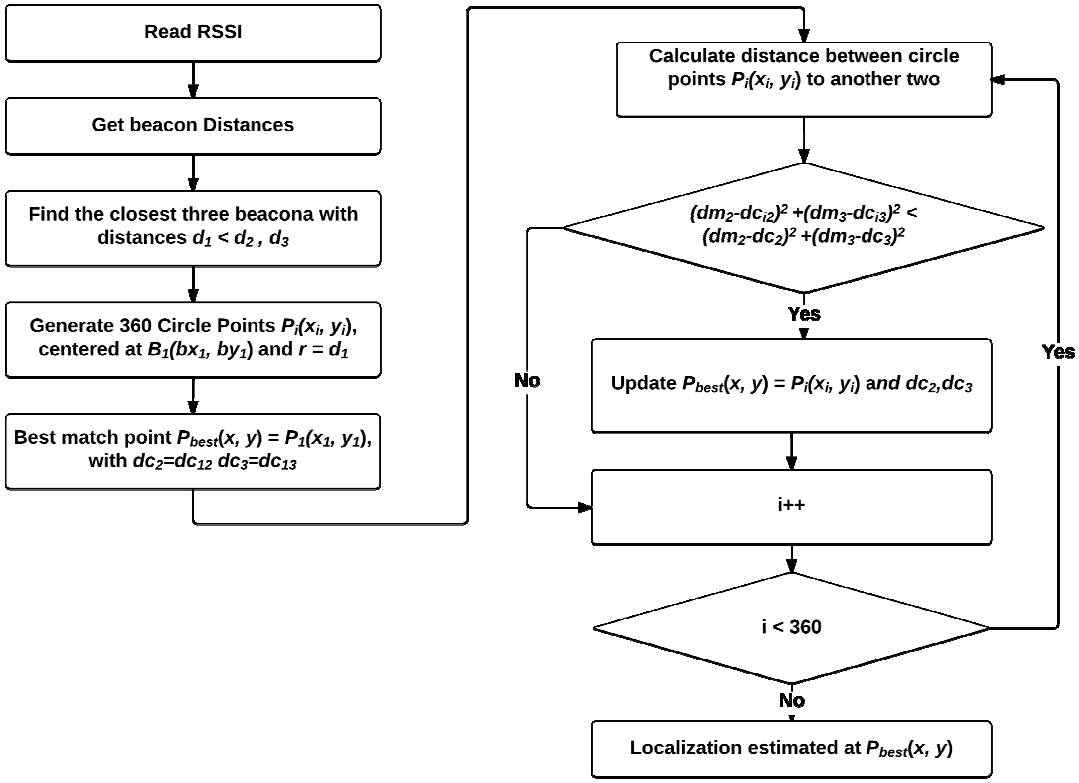


Figure.46. Flowchart of Trilateration

4.4.2. Particle Filter

The Monte Carlo filter, which is usually referred to particle filter, is first presented in [39]. In particle filter, the position of the mobile node is represented as probability densities of position states by a set of random samples. A combination of various observable variables such as measurements to the references, previous estimated position, and orientation of mobile node and the movement of mobile node are used to build and update the states.

Particle filter is consisted of two phases: position prediction phase and position update phase. During the position prediction phase, a movement model is used to describe the transitions between different states. While in the update phase, a measurement model is used to describe the characteristics of the sensors.

Formally speaking, the key idea of particle filter is to get the posterior $p(x_k|z_{1:k})$ given the sensor measurement and prior estimation $p(x_k|z_{1:k-1})$. The prior estimation could be represented as:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}, z_{1:k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \quad (15)$$

where $\int p(x_k|x_{k-1}, z_{1:k-1})$ is the conditional PDF of any state x_k at time k given the previous state x_{k-1} and the history of measurements $z_{1:k-1}$ from time 1 up to time k – 1. And $p(x_{k-1}|z_{1:k-1})$ is the previous state x_{k-1} given the measurement history $z_{1:k-1}$.

If a Markov model of order one is assumed, which means the current state only depends on the previous one state and the next state only depend on current state, then $p(x_k|x_{k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$. So Formula(15) becomes:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \quad (16)$$

where $p(x_k|x_{k-1})$ can be interpreted as the motion model.

So the posterior becomes:

$$p(x_k|z_{1:k}) = \alpha_k p(z_k|x_k)p(x_k|z_{1:k-1}) \quad (17)$$

where $p(z_k|x_k)$ is the conditional PDF of any measurement z_k at the time k given the state x_k at the same time, which is also called likelihood and could be interpreted as the sensor measurement model. α_k is the normalization constant $\frac{1}{p(z_k|z_{1:k-1})}$.

An algorithm based on sampling importance resampling (SIR) particle filter is represent in Table.9.

Algorithm PARTICLE_FILTER(M, LANDMARKS_SET)

1. Initialize particle filter with M particles, $w_0^{(i)} = 1/M$, where i = 1,…,M.
2. At time *k*, predict particles states $x_k^{(i)} \sim p(x_k|x_{k-1})$based on the motion model for i = 1,…,M
3. Calculate the importance factor, which is the weight $w_k^{(i)}$ according to $p(z_k|x_k)$
4. Normalize the weights as $\widetilde{w}_k^{(i)} = w_k^{(i)} / \sum_{j=1}^{M} w_k^{(j)}$
5. Resample the particle set according to $\widetilde{w}_k^{(i)}$, building a new particle set with the same size of M
6. Repeat Step 2 to 5 as for time k = 1,2,3…

Table.9. Particle Filter Algorithm

Generally speaking, at the beginning of particle filter algorithm, thousands of particles are randomly and uniformly generated to cover the whole indoor area. Since there is no sensor input and no observation from references, each particle is only an initial guess of the current position of the unknown mobile node, with equal probability. As for the prediction phase, when the mobile node moves, all the particles would move correspondingly. This is followed by update phase. For each particle, the actual position is known. So the distances between particles and each of the reference beacons can be calculated. Then the differences between measured distances from mobile node and the calculated distances from particles are used to generate a weight, which is assigned to each particle. As shown in Figure.4, it is clear that particle 1 has a larger difference than that of particle 2. As a result, a lower weight is assigned to particle 1. In Figure.4, it uses 3 references beacons, which makes this step look like trilateration. But the number of reference beacon is not necessary 3 and can be arbitrarily more. After updating weights on all of the particles, a weight normalization process will be performed to make the entire weights sum to 1.

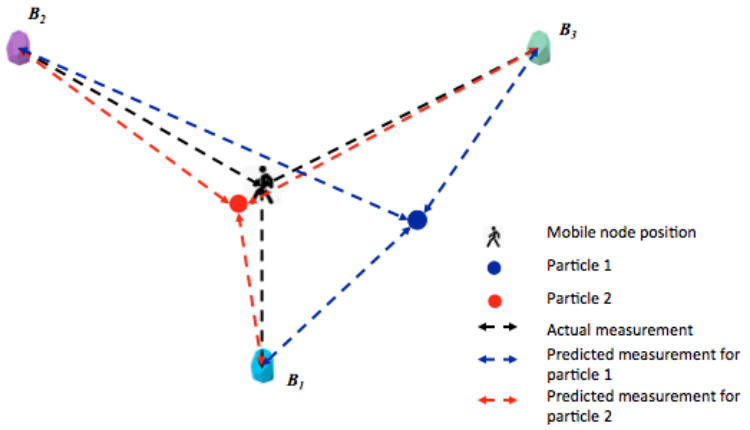
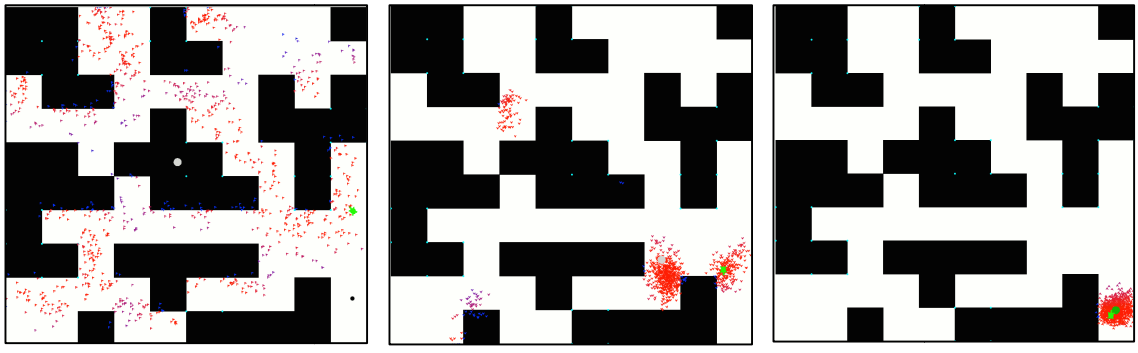


Figure.47. Weight Update

The final but the most important process is the resampling process. Resampling process transforms a particle set of size M into another particle set of the same size. Before the resampling process, the predicted position of the mobile node is generated from the state that has only implemented motion model. During the resampling process, particles are chosen from the previous set to a new set according to their weights. That is to say, in Figure.47, particle 2 is more likely to be chosen than particle 1 since it has a higher weight. A single particle can be sampled multiple times to the new set. After the resampling process, the predicted position of the mobile node is successfully updated according to current sensor model.

The following figures shows how particle filter usually works:

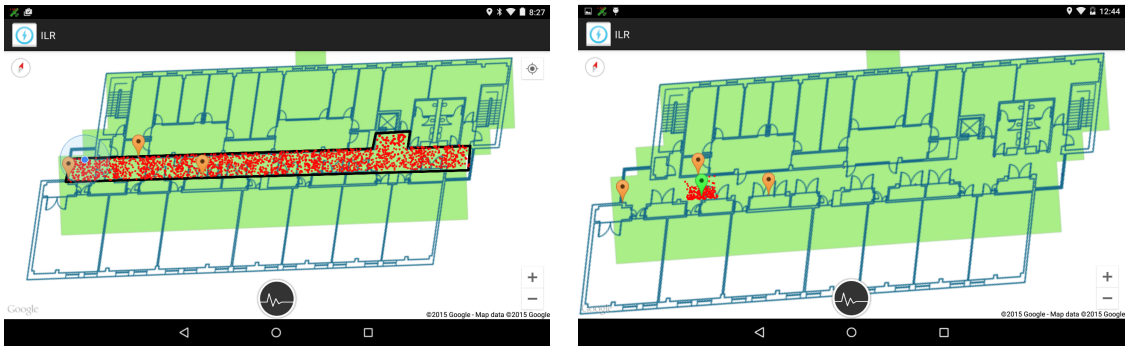


(a) Initialize **(b) Update** **(c) Result**

Figure.48. Particle Filter Demo

As shown in Figure.48(a), thousands of particles are generated randomly at the initial stage. The green dot indicates the true position of the mobile node and the grey node indicates the estimated position of particle filter. As the mobile node moves, the particles form several clusters based on observables variables, shown in Figure.48(b). Eventually, as shown in Figure.48(c), only one cluster will be formed and the predicted position is very near the true position.

In my application, the sensor model is built based on the propagation model that discussed in Section.4.2. Initially, 2000 particles are generated to cover the whole localization area, as it is shown in Figure.49(a). Due to the huge noise from the compass in indoor environment, the orientation is generated randomly. So the motion in this case follows Brownian motion model. As a result, the motion model predicts a person can move to any direction at any time. This model is shown to work well in particle filter in [36]. As the same as the trilateration method, only three closest beacons will be chosen as the reference. So the mobile node collects RSSI from these beacons and calculates the measurement distance. Combined with the calculated distances for each particle, weight for the particle is determined. After the resample process, only one cluster will be formed and the position of the mobile node is chosen as the average position of all the particles, as it shown in Figure.49(b). The overall flowchart of the implemented particle filter localization algorithm is shown in Figure.50.



(a) Initialization **(b) Estimate position**

Figure.49. Particle Filter in Android Application

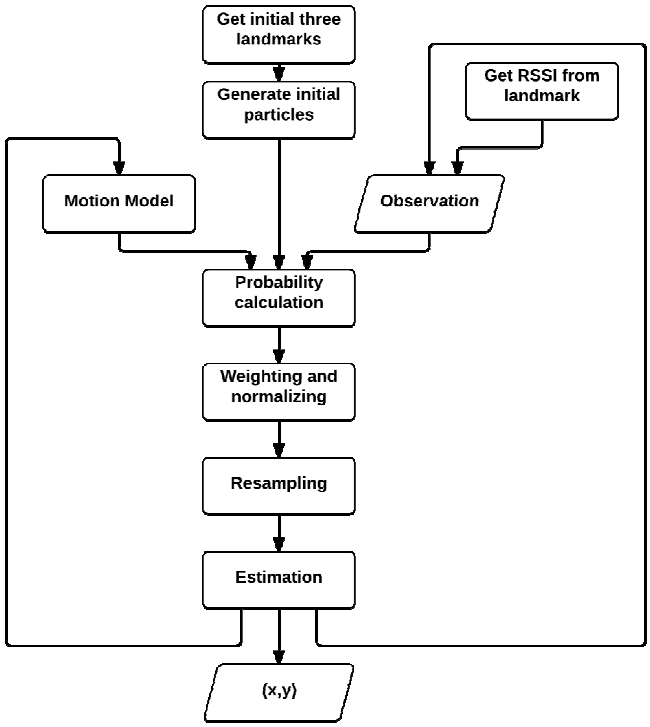


Figure.50. Flowchart of Particle Filter Localization Algorithm

4.4.3. Hybrid of Trilateration and Particle Filter

In this thesis, a hybrid approach is proposed, which combines the localization results of trilateration and particle filter. As we discussed in the previous sections, the localization result of trilateration replies more on the closest beacon, which reduces the bias from the other farther beacons. But it cannot reduce the bias from the closest beacon. While in the particle filter approach, although it can tolerant some noise, the localization result is not good enough when the mobile node is very near a beacon. Therefore, the localization of this hybrid approach is defined as the linear combination of the results from trilateration and particle filter, which is shown in Formula.(18):

(18)

where is the localization result of the hybrid approach. and are the results of trilateration and particle filter, respectively. and are the weights of the two localization results and .

In the current implementation, the values of w_{tri} and w_{pf} are chosen arbitrarily according to the RSSI reading. That is to say, there is an RSSI value, which serves as a threshold to indicate whether we should believe more on the trilateration result or particle filter result. After multiple tests, the threshold is chosen as -66, which indicates the unknown node is about 1 meter away. Therefore, the values of w_{tri} and w_{pf} are chosen as:

	RSSI ≥ -66	RSSI < -66
w_{tri}	0.7	0.35
w_{pf}	0.3	0.65

Table.10. Weights selection in hybrid approach

So to sum up, the hybrid localization is defined in Formula(19):

$$Loc_{hybrid} = \begin{cases} 0.7Loc_{tri} + 0.3Loc_{pf}, & RSSI_{threshold} \geq -66 \\ 0.35Loc_{tri} + 0.65Loc_{pf}, & RSSI_{threshold} < -66 \end{cases} \quad (19)$$

4.4.4. Localization Improvements

As we discuss in section, there are a lot of factors influence the RSSI readings from BLE beacons. While all of the three localization algorithms discussed in the previous section reply highly on the accuracy of RSSI readings. So several improvements are proposed in this section to improve the localization precision.

4.4.4.1 Neighbor Beacon

For each beacon that will be used for localization, it maintains a set of neighbors, which are geographically closest n beacons that are also used for localization. The neighbors can be assigned during the deployment phase. But it can also be calculated in the localization app. In my application, two neighbors are assigned to each beacon. Therefore, when performing the localization, the application would first locate the nearest beacon, which is the anchor beacon, and then get its two neighbor beacons. It can be guaranteed that the true position of the unknown node is closer to the three beacons than other beacons. Even if sometimes it would receive a larger RSSI from a fourth beacon, which is not a neighbor to current anchor beacon, the

application would treat such reading as unstable reading and ignore it. For example, in Figure.51, $Beacon_1$ is regards as the anchor beacon. Although $Beacon_4$ has a higher RSSI reading than $Beacon_3$, the application would still use the reading from $Beacon_3$ instead of $Beacon_4$ for the reason that $Beacon_4$ is not a neighbor.

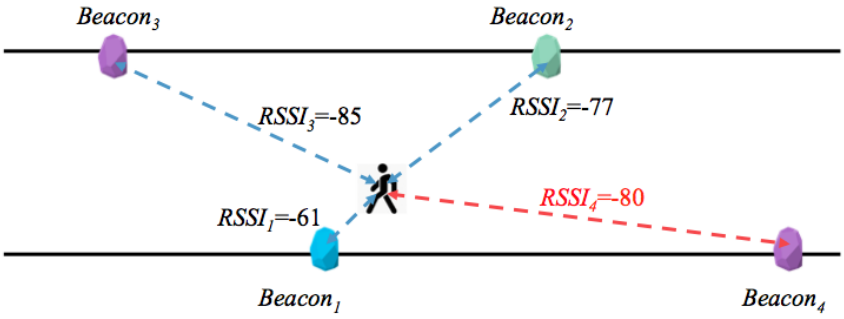
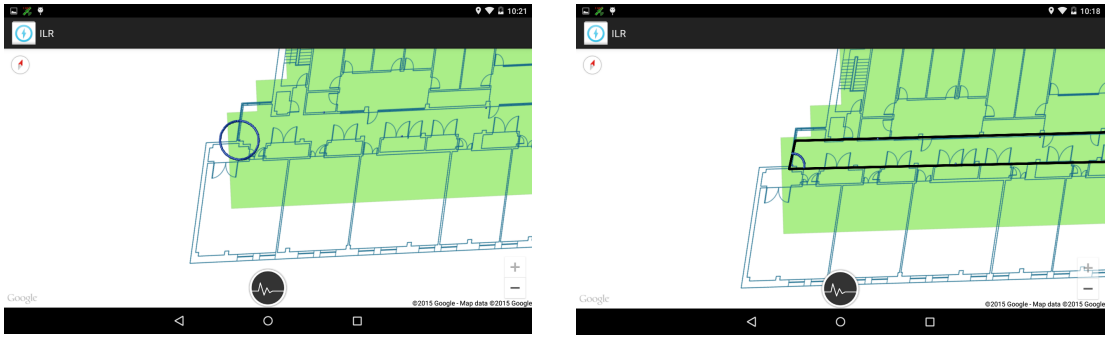


Figure.51. Neighbor Beacons

4.4.4.2 Valid Localization Area

During the implementation, a valid localization area concept is also integrated. A valid localization defines the possible locations for the localization area, distinguishing itself from unrealistic localization position, such as inside a cabinet or outside the building. This is helpful for both trilateration and particle filter algorithm. As we can see from Figure.52(a), before using the valid localization area, the algorithm would make points outside the building as candidates for predicting the position of the unknown node. After integrating valid localization area, only a portion of points, which are inside the building, would be used to get the predicted position.



(a) Before **(b) After**

Figure.52. Effect of Valid Localization Area

Also, as we have seen in Figure.49(a), using a valid localization area would constrain all the particles be initialized legitimately. When the particles move and update, all the particles are guaranteed to be a valid possible state of the unknown mobile node.

4.4.4.3 Jumping Prevention

Due to the unstable RSSI readings from beacons, the localization result would jump to different locations. Given whether the unknown node is moving or not, the jumping can be categorized into fixed point jumping and mobile jumping.

As shown in Figure.53, predicted position can vary a lot within a short time period. In some cases, the predicted position would jump to a place where is quite far away.

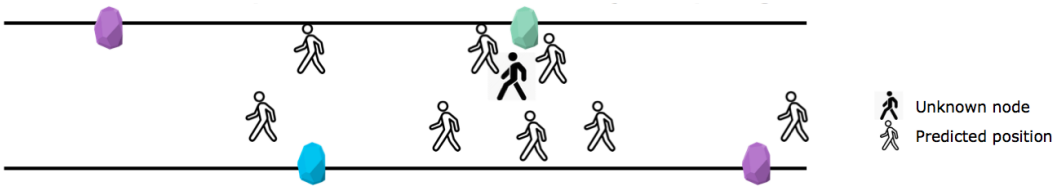


Figure.53. Fixed Point Jumping

To solve this problem, a fixed point jumping tolerant area is integrated. As shown in Formula.(20) and Figure.54, if predicted location at time t is no farther than 2.5 meters from the predicted localization at time t-1, then it is regarded as reasonable jumping. Otherwise, the prediction location will be neglected.

(20)

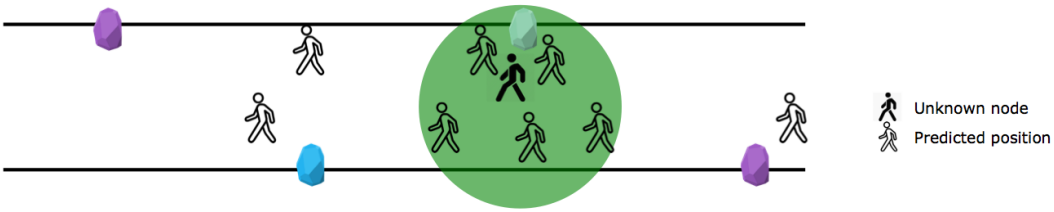


Figure.54. Fixed Point Jumping Solution

The jumping problem also exists when the unknown node is moving. As it is shown in Figure.55, the distance between localizations at t_0 and t_1 is greater than 8 meters while $t_1 - t_0 = 1$s,

which in turn indicates the speed of human is greater than 8m/s. Considering this application is not used in a sprint or fast moving scenario, such a moving speed is illegitimate.

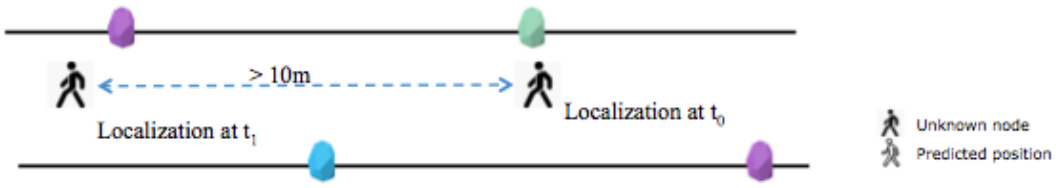


Figure.55. Mobile Jumping

The solution is straightforward. As it is shown in Figure.56, several interpolated positions are added between the localizations at t_0 and t_1, so that the localization looks much smoother when the unknown node is moving.

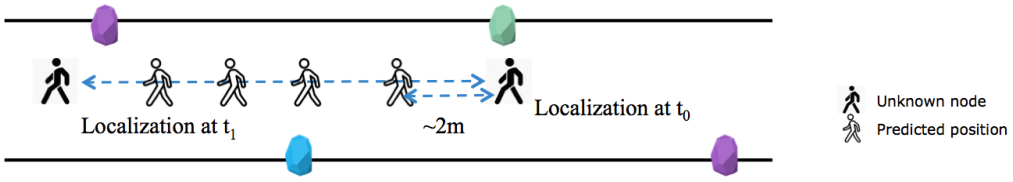


Figure.56. Mobile Jumping Solution

4.4.5. Localization Test

Based on the three localization approaches as well as the improvement methods, multiple tests have been carried out and evaluated.

4.4.5.1 Test Bed

As it is discussed in the previous chapters, the current propagation model and the beacon deployment strategy are for a corridor-like environment. So the primary scenario is chosen as the corridor in the third floor of the Knowles Engineering Building. The blueprint and a photo of environment are shown in Figure.57(a) and Figure.57(b).

(a) Blueprint **(b) Photo of the Environment**

Figure.57. Test Bed for Indoor Localization and Navigation

The length of the corridor is about 50 meters and the width is about 2 meters. One side of the corridor consisted of multiple recessed doors and the other side is consisted of large portion of glass wall. With different structures and construction materials, the noise of RSSI is negligible. 5 test points are chosen as it shown in fig. A Moto X Android smartphone is used. 100 data are collected for each of the localization approaches at every test point, and the smartphone should be held still without many movements in hand.

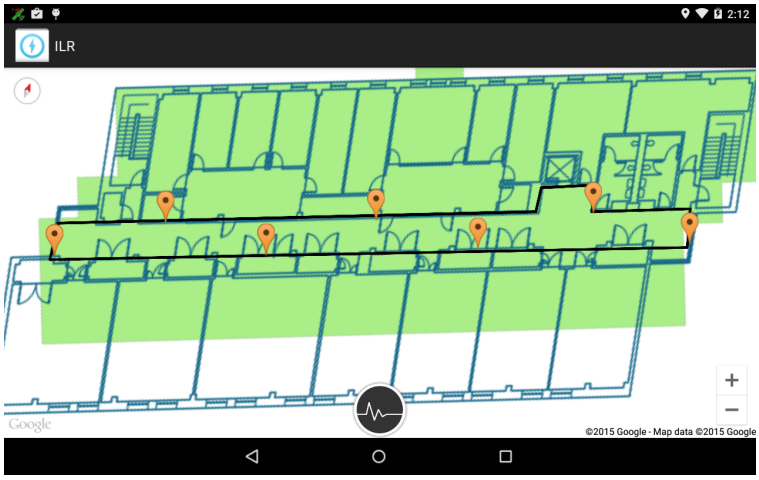


Figure.58. Beacon Deployment at Test Side

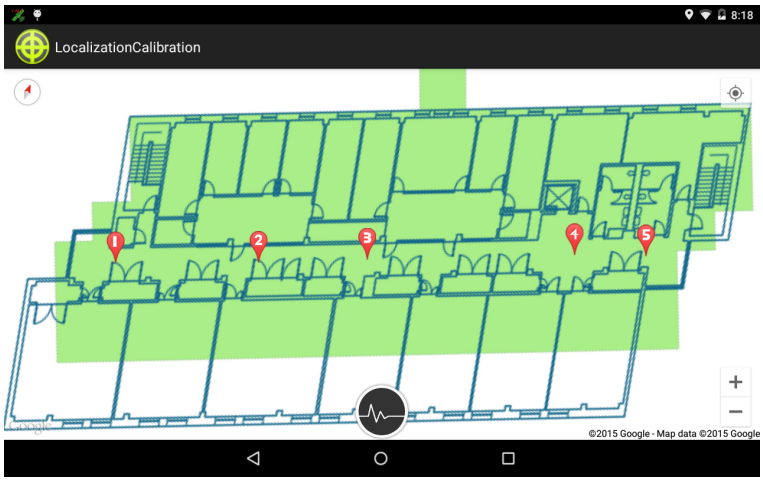


Figure.59. Test Points

4.4.5.2 Test Results

The localization error is evaluated as the physically distance between the test point and the localization prediction at that point. The mean errors of the three approaches at five test points are shown in Figure.60.

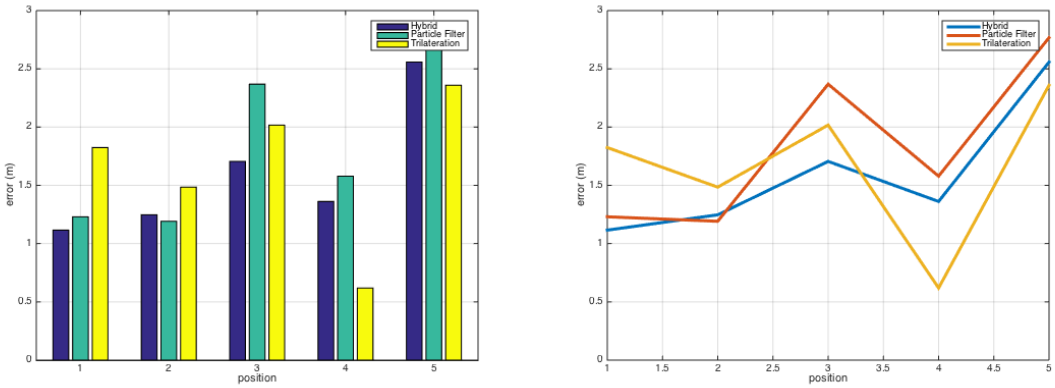


Figure.60. Mean Error at Five Test Points

In addition to the mean error at each point, the variance of localization error, as well as the least and worst error for each approach and position are also shown in Table.11 and Table.12. As it can be concluded, the localization accuracy varies by different localization algorithms and different test points. The mean error for each of the approach is within 3 meters. For the worst case, the error is within 7 meters, which is acceptable in the indoor search and research scenario.

Algorithm	Hybrid		Particle Filter		Trilateration	
Test Point	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
1	1.115	0.494	1.230	0.719	1.824	0.192
2	1.247	0.7073	1.191	0.641	1.484	0.639
3	1.705	1.311	2.368	1.355	2.017	2.147
4	1.361	0.749	1.578	0.869	0.619	0.540
5	2.558	1.317	2.767	1.635	2.359	1.037
Total	1.597	1.102	1.827	1.276	1.66	1.275

Table.11. Mean and Standard Deviation of Localization Error

Algorithm	Hybrid		Particle Filter		Trilateration	
Test Point	Max	Min	Max	Min	Max	Min
1	2.123	0.518	2.700	0.241	2.216	1.393
2	2.206	0.037	2.402	0.072	3.187	0.562
3	5.518	0.501	5.491	0.958	6.472	0.581
4	3.624	0.162	4.484	0.248	2.602	0.064
5	5.429	0.415	6.553	0.538	5.394	1.287

Table.12. Max and Min of Localization Error

So from the above data, it is safe to say the proposed hybrid approach out-performs both particle filter and trilateration.

In order to take a look at the localization latency, the first 20 localization errors of the hybrid approach is shown in Figure.61. As we expected, the first several localization results have higher errors. The errors drop down as the user stays longer.

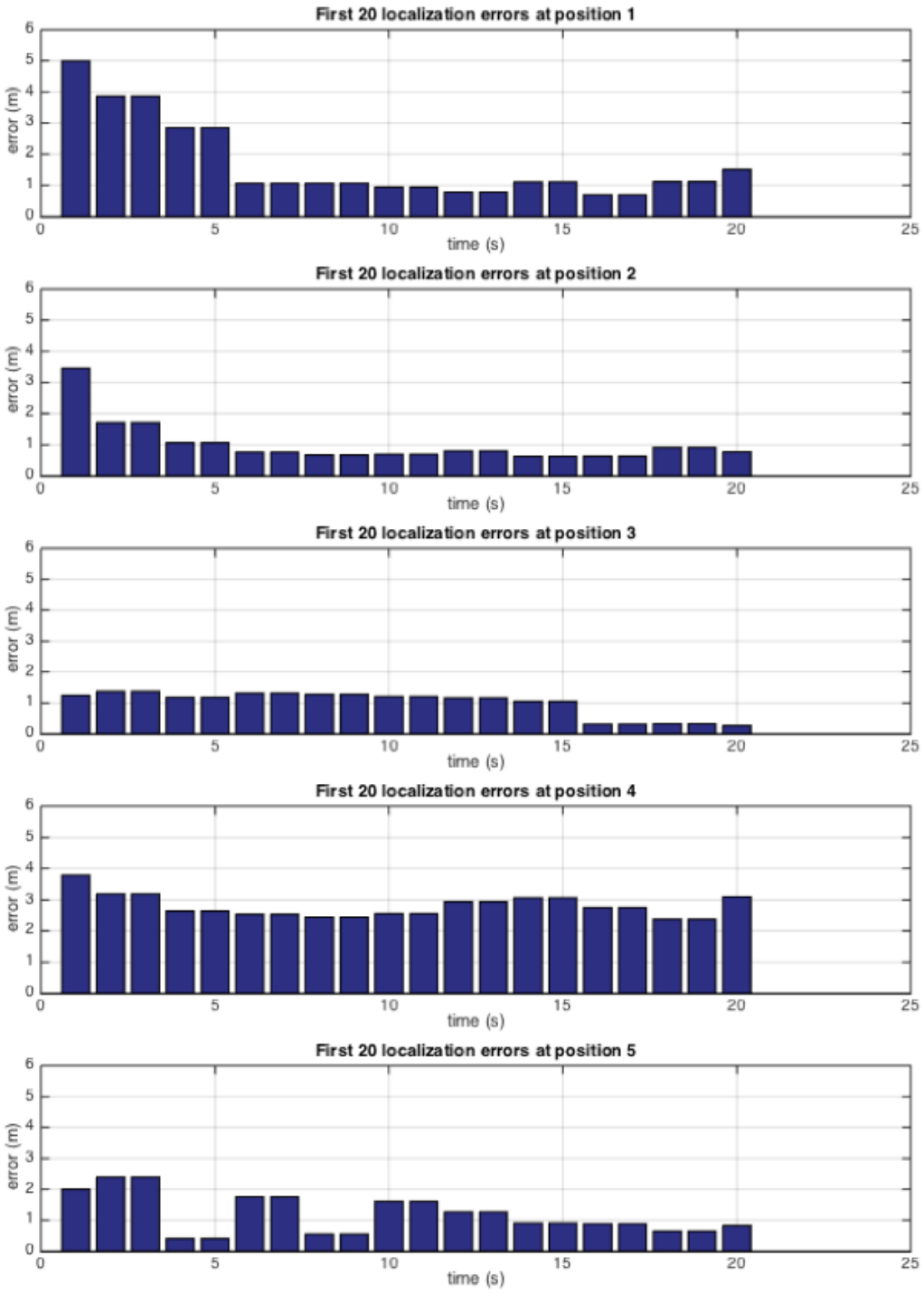


Figure.61. Localization Error of First 20 Predictions at 5 Test Points

And the overall error distribution is shown in Figure.62.

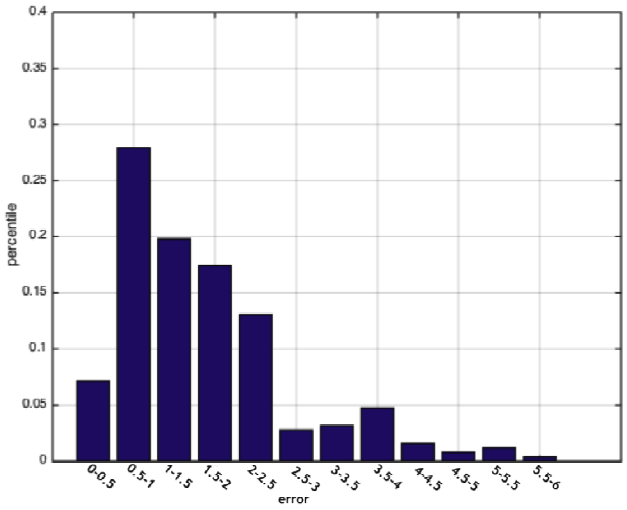


Figure.62. The Overall Error Distribution

Y-axis specifies the percentile of error locates in each interval. As we can see, more than 70% of errors are less than 2 meters and only around 12% are larger than 3 meters.

Another test is also designed to investigate the localization performance when the unknown node is close to and far from the beacon. More specifically, according to the deployment plan, a close scenario is defined when the distance is within 1 meter. While in a far scenario, the distance is about 4 meters. For each scenario, 100 data are collected for each of the localization approach. The result is shown in Table.13. Five scales are defined as follows: Excellent: <1m, Very Good < 2m, Good < 3m, Fair < 5m and Poor > 5m.

Algorithm	Hybrid		Particle Filter		Trilateration	
Performance	Near	Far	Near	Far	Near	Far
Excellent	98%	20%	61%	10%	100%	4%
Very Good	100%	43%	97%	63%	100%	19%
Good	100%	100%	98%	74%	100%	80%
Fair	100%	100%	100%	98%	100%	96%
Poor	0	0	0	2%	0	4%

Table.13. Localization Performance for Near and Far Scenario

The results prove again the closer the distance between transmitter and receiver, the better the localization prediction will be. Also, trilateration algorithm outperforms particle filter in the close scenario while underperforms in the far scenario.

We can conclude that in this corridor, the fixed-point localization is pretty good. The mobile localization accuracy may vary from the above result, which is not shown in the above figures or tables. Since an averaged RSSI value is used, there can be latency of one to three seconds during the mobile localization. The mobile localization result can be seen in the navigation section.

4.4.5.3 Additional Test at An Open Space Area

An additional test is also designed in order to test the robust of the localization approaches in a different environment. In this test, 8 beacons are deployed in the Gunness Student Center, which is an open space area, as shown in Figure.63. As we discussed in Section 4.2, the propagation model might differ from the one used in the previous corridor environment. However, in order to test how would the propagation model affect the localization result, a same model is used here. In addition, no valid localization area is set.

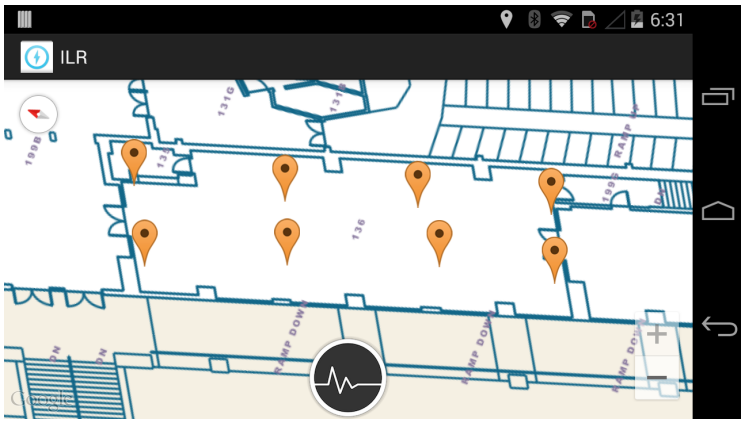


Figure.63. Test Bed 2

Five test points are also chosen, as shown in Figure.64. As the same as the previous tests, user holds the smartphone at the test point without any movement.

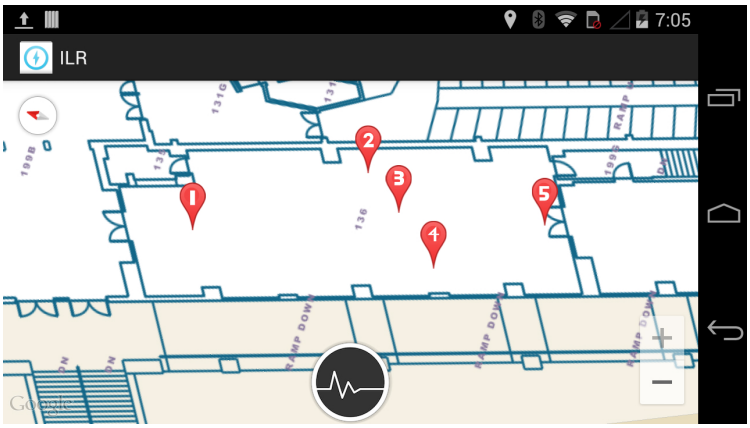


Figure.64. Test Point for Test Bed 2

The mean localization errors at each point are shown in Figure.65.

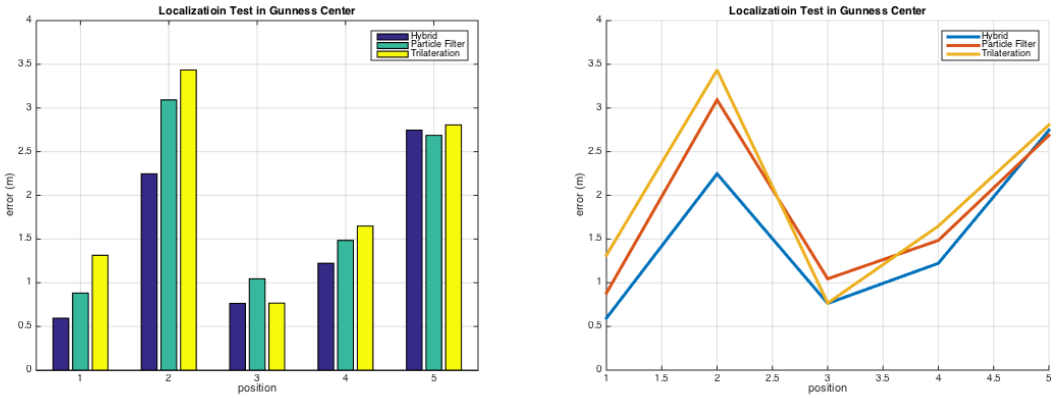


Figure.65. Mean Localization Error for the Five Test Points in Gunness Student Center

Table.14 shows the overall localization performance of the methods.

Algorithm	Hybrid	Particle Filter	Trilateration
Error Mean	1.515	1.838	1.994
Standard Deviation	0.208	0.577	0.919
Max Error	3.672	6.675	9.275
Min Error	0.148	0.455	0.530

Table.14. The Localization Error in Gunness Student Center

From the above results, we can conclude that the fixed-point localization performance is acceptable in a different environment, even if the propagation model remains the same.

4.5 Indoor Navigation

Navigation in an indoor environment during a search and rescue process is extremely important. Unlike an outdoor environment, an indoor environment is consisted of corridor, rooms and doors, which makes it much more complex. An effective navigation not only provides rescuer a route to find victims, specific rooms or exits quickly, but also gives rescuer a better understanding of the indoor environment.

From the localization algorithm described in the previous section, the position of rescuer is determined. Combining the blueprint of the indoor environment, navigation becomes achievable.

In this thesis, A* path finding algorithm is used to quickly find the optimum path between the rescuer's position and the selected destination.

4.5.1. A* Path Finding Algorithm

A* is a search algorithm that is widely used in path finding and graph traversal. It is first described by Peter Hart, Nils Nilsson and Bertram Raphael in [40]. As an extension for Djikstra algorithm, A* achieves better performance by using heuristic.

Formally speaking, A* uses a best-first search and find a path with least cost from initial point to the destination point.

$$Cost(x) = g(x) + h(x) \tag{20}$$

As shown in Formula.(20), the cost function of A* is consisted of two parts:

- an heuristic estimate of the distance from the destination to the current point
- the moving cost from the initial point to the current point.

Further more, is usually said to be admissible heuristic, meaning that is should not overestimate the distance. So is usually chosen to be the straight line distance to the destination.

The overall path finding process of A* is shown in Figure.66 to Figure.67.

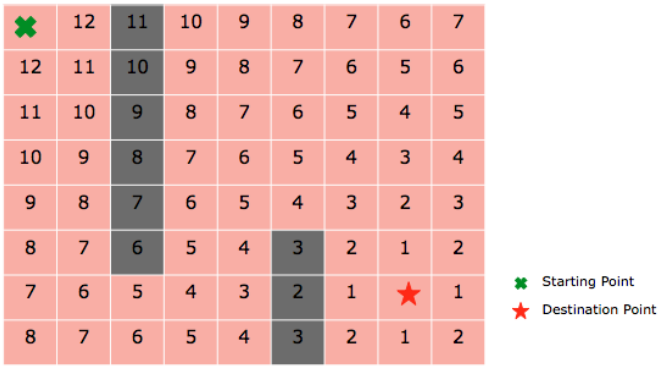


Figure.66. Heuristic Estimate

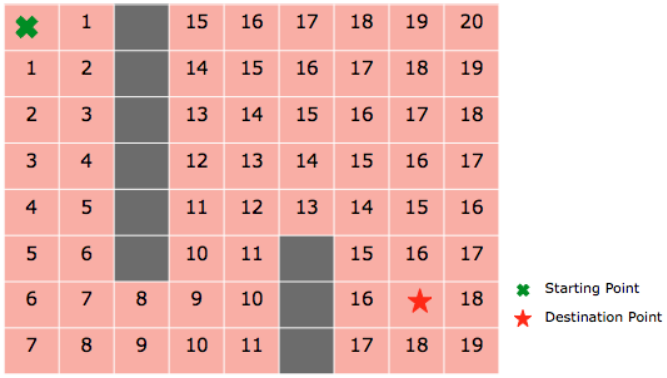


Figure.67. Moving Cost

Figure.66 shows the heuristic cost of this grid-based environment. It is important to notice that the heuristic value is not affected by any obstacles. While in Figure.67, the cost from starting point to current point is calculated. This time, obstacles must be taken into consideration. Combining the heuristic cost and moving cost, it is easy to get the best path shown in Figure.68.

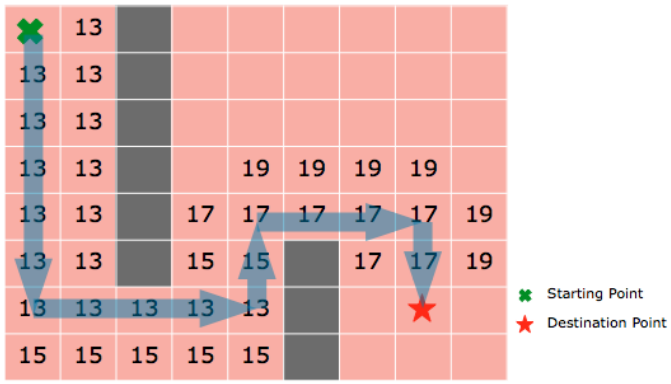


Figure.68. Path Generated by A*

As we can see in Figure.68, when A* traverses the graph, it keeps a priory queue of alternate path segments along the way and it only takes the path with lower cost.

Figure.69 shows the path finding in the same graph using Djikstra. It is guaranteed to find the shortest path, but on the other hand, it searches 10 more nodes to find the destination. That is why A* is usually out performs Djisktra in terms of efficiency.

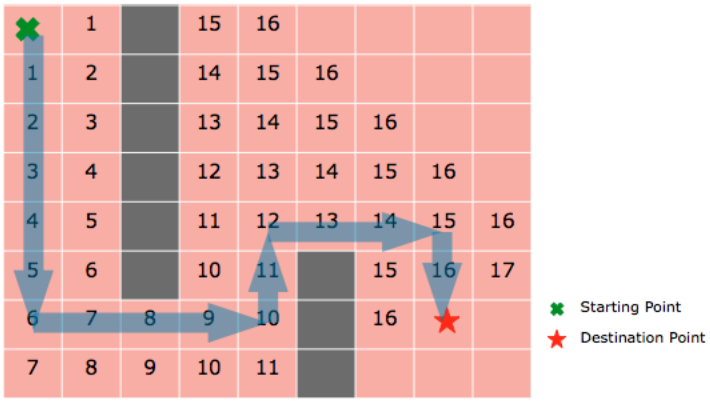


Figure.69. Path Generated by Djikstra Algorithm

4.5.2. A* Implementation

As the same as localization, the navigation in this thesis is also focused on the corridor-like indoor environment.

As shown in Figure.70, the corridor is modeled with more than 30 grids. Each black dot on the map represents the center of the grids. Multiple rooms are along the corridor and an exit is

at each end of the corridor. A graph represented by grids and landmarks is defined in the application.

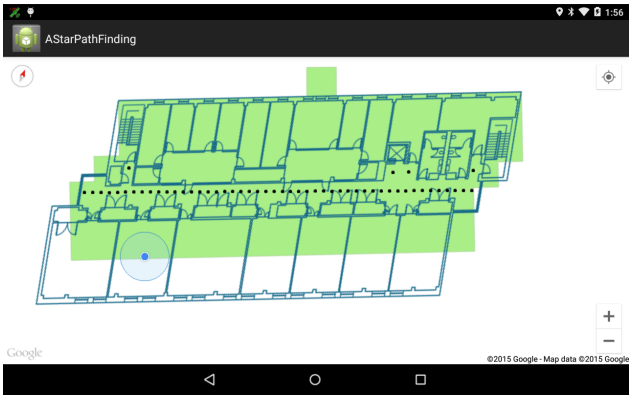


Figure.70. Environment Modeling for A*

The heuristic value is chosen as the geographical distance between two points. The current localization is determined by the result of hybrid localization method. The destination is chosen from the sliding panel on the left of application, as shown in Figure.71.

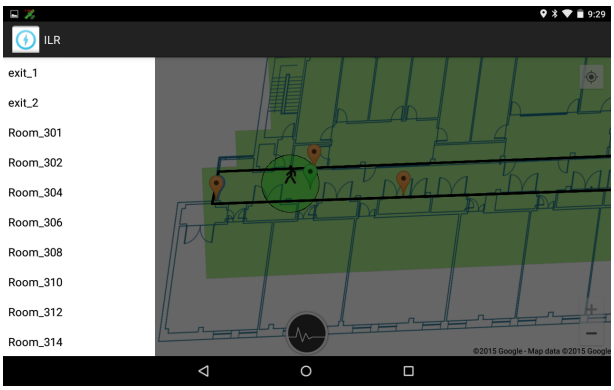


Figure.71. Destination Selection

Once the destination and the current location are determined, a suggested shortest path to the destination will be shown on the screen. Figure.72 and Figure.73 show the paths to exit and rooms.

Figure.72. Path to Exit1

Figure.73. Path to Room 309E

While rescuer moves, the path changes accordingly. In other word, if the rescuer takes a wrong way, then a re-route will be automatically carried out and generate a new path.

A comparison between the actual moving path, which is generated based on the hybrid localization algorithm, and the planed path after one navigation is shown in Figure 74 and Figure 75. As we can see, localization and navigation cooperate pretty well and the user can successfully find the destination.

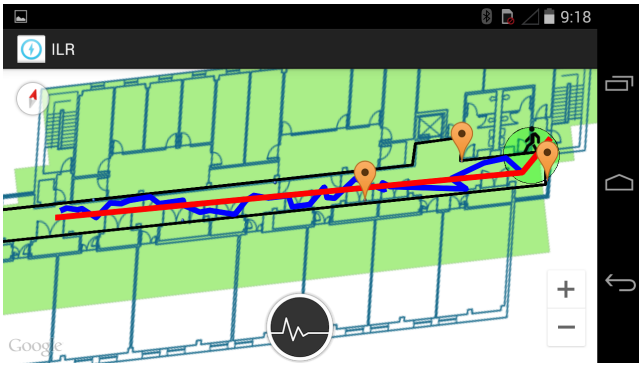


Figure.74. Navigation Path and the Actual Moving Path to Exit 1

Figure.75. Navigation Path and the Actual Moving Path to Room 302

CHAPTER 5

FUTURE WORK

This chapter describes possible future modifications and improvement to the current DIORAMA mobile platform and the indoor localization and navigation system.

5.1 Future Work on DIORAMA Mobile Platform

5.1.1 Obstacle Avoidance Algorithm

Current obstacle avoidance strategy is based on predesigned motion pattern. It works quite well in an open space area, where there are not so many obstacles. But if the robot goes into a maze-like area or an indoor environment, where the robot might be surrounded by obstacles. In that case, using the predesigned motion pattern can possibly make the robot stick in the middle of these obstacles. Also, there is no way to predict the location of obstacle, so a predesigned motion pattern is not a universal solution.

There are a lot of existing obstacle avoidance strategy, such as A* path finding algorithm and D* incremental search algorithm [41]. Without the need of intense computation resource, it is possible to integrate these algorithms into the current on-broad robot application. Based on the sensor information, it is possible for the robot to find an alternative route in a short time.

5.1.2 Automatic Area Assignment

The current patron way points and destinations are assigned by incident commander manually. In order to increase the automaticity of the mobile platform, an automatic collaborative patron area generation mechanism is one of the candidates.

Based on the RSSI signals received by human responders or other mobile platforms, the commander application would know automatically about which areas are currently covered and which are not. Then patron command and moving path would automatically send to the nearest mobile platform.

5.2 Future Work on Indoor Localization and Navigation System

5.2.1 Fast Beacon Deployment

The current fast deployment strategy only provides guideline for corridor like environment. As for large open space, a more automatic deployment algorithm is needed. A hexagon deployment strategy is a good candidate.

At the same time, in the current system a threshold is added aiming to decrease the RSSI signal noise, but the effect is limited. As it is shown in [42], an adaptive or self-calibration propagation model might be a better option. Also, although the same propagation model is used for testing two different environments and achieved pretty good localization accuracy, it is better to build separate propagation model for each kind of environment or even each environment. As we discussed in Section 4.2, the current propagation model is built based on large set of data in an offline phase. However, it is possible to migrate all the steps into smartphone, making the process more dynamic and flexible.

5.2.2 Motion Model in Particle Filter

The motion model used in the current particle filter localization algorithm is the Brownian motion model, indicating that the motion of people is unpredictable. However, in [43], the author tried to build motion model based on different criteria. This is a good direction to be studied in order to improve the localization accuracy of particle filter.

5.2.3 Orientation Issue

None of current localization algorithms take orientation into consideration. However, besides the physically location, orientation is also an important aspect in localization. Although most Android smartphones are integrated with orientation sensor, the orientation accuracy is so poor due to the huge noise in the indoor environment. A more advanced sensor fusion mechanism is a prerequisite to get a good orientation result.

Once reliable orientation information is achievable, the localization accuracies of particle filter as well as the hybrid algorithm are expected to improve.

CHAPTER 6

CONCLUSION

In this thesis, I introduced DIORAMA mobile platform for outdoor real time information collection. The autonomous mobile platform was able to successfully collect RSSI data as well as real time video image. A client server structure is designed to control and monitor the robot remotely. Different control methods are used to control the robot, enabling the robot move to waypoints and destination as expected. Google Direction API and visibility graph re integrated into the commander application, which could generate a shortest path for the robot to cover all the assigned area. Also, as an essential part, obstacle avoidance and immobility detection are also taken into consideration to make the robot movement smoother. Finally, multiple tests has been designed and performed. It showed that the mobile platform was able to move to waypoints and destinations automatically and collecting expected real time data at the same time.

I also proposed a system for indoor localization and navigation for responders. Real time localization data of responder is collected based on Android smartphone and BLE beacons. A RSSI propagation model is built for a corridor-like environment and a fast deployment strategy is discussed. This enables rescuers quickly deploy the BLE beacons and use them for indoor localization and navigation in a corridor-like environment without pre-deployment. Trilateration and particle filter are implemented for localization. A hybrid approach combining these two algorithms has also been proposed. By integrating improvement methods, the accuracy of the localization is increased. Based on multiple tests, the localization errors are mostly within 3m, which is pretty good for indoor search and rescue. Finally, a navigation system based on A* is also developed in the system. This gives responder a fast route to specific rooms or exits. Reroute is also integrated when responder takes a wrong way.

BIBLIOGRAPHY

[1] J. Mistovich, Brady Prehospital Emergency Care Sixth Edition, pg, 866.

[2] A. Ganz, J. Schafer, X. Yu, G. Lord, J, Burstein, G. Ciottone, "Real-time Scalable Resource Tracking Framework (DIORAMA) for Mass Casualty Incidents", International Journal of E-Health and Medical Communications, 4(2), 34-49, April-June 2013.

[3] A. Ganz, J. Schafer, Z. Yang, J. Yi, G. Lord, and G. Ciottone, "Mobile DIORAMA-II: Infrastructure less Information Collection System for Mass Casualty Incidents", 36th Annual International Conference of Engineering in Medicine and Biology Society (EMBC), 2682-2685,26-30 Aug 2014.

[4] M. Kjaergaard, H.Blunck, T.Godsk, "Indoor positioning using GPS revisited", T.Toftkjaer, D.Cristensen, K.Gronbaek, Pervasive'10 Proceedings of the 8th international conference on Pervasive Computing, 17-20 May 2010.

[5] K. Cheung, S. Intille, K. Larson, "An inexpensive Bluetooth based indoor position hack".

[6] H. Sugiyama, T. Tsujioka, M. Murata, "Coordination of Rescue Robots for Real-Time Exploration Over Disaster Areas", 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 170 – 177, 5-7 May 2008.

[7] K. Ryu, "Autonomous Robotic Strategies for Urban Search and Rescue", Ph.D. dissertation, Dept. Mechanical. Eng. Virginia Tech, Blacksburg, VA, 2012.

[8] H. Kuntze, C.Frey, I. Tchouchenkov, B. Staehle, E. Rome, K. Pfeiffer, A. Wenzel, J. Wöllenstein, "SENEKA - Sensor Network with Mobile Robots for Disaster Management", IEEE Conference on Homeland Security Technologies (HST), 406-410, 13-15 Nov 2012.

[9] J. Freeman, V. Amrita, V. Omanan, M. Ramesh, "Wireless Integrated Robots for Effective Search and Guidance of Rescue Teams", Eighth International Conference on Wireless and Optical Communications Networks (WOCN), 1-5, 24-26 May 2011.

[10] G. Tuna, V. Gungor, K. Gulez, "An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters", Ad Hoc Networks, Volume 13, Part A, Pages 54–68, February 2014.

[11] T. Kovács, A. Pásztor, Z. Istenes, "A multi-robot exploration algorithm based on a static Bluetooth communication chain", Robotic and Autonomous Systems, Volume 59 Issue 7-8, July 2011

[12] M. Norouzi, M. Yaghobi, M.R. Siboni, M. Jadaliha, "Recursive Line Extraction Algorithm from 2D Laser Scanner Applied to Navigation a Mobile Robot", IEEE International Conference on Robotics and Biomimetics, 2127-2132, 22-25 Feb 2009

[13] M. Norouzi, A. Karambakhsh, M. Namazifar, B. Savkovic, "Object Based Navigation of Mobile Robot with Obstacle Avoidance using Fuzzy Controller", IEEE International Conference on Control and Automation, 169 – 174, 9-11 Dec. 2009

[14] M. Brunner, B. Bruggemann, D. Schulz, “Motion Planning for Actively Reconfigurable Mobile Robots in Search and Rescue Scenarios”, 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR),1-6, 5-8 Nov. 2012

[15] C. Fischer, H. Gellersen, “Location and Navigation Support for Emergency responders- A Survey ", Pervasive Computing, IEEE, Volume 9, Issue 1, pg 38-47, 2009

[16] J. Wilson, V.Bhargava, A. Redfern, P. Wright, “A wireless sensor network and incident command interface for urban firefighting”, Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, 1-7, 6-8 Aug. 2007

[17] M. Klann, M.Geissler, “Experience Prototyping - A New Approach to Designing Firefighter Navigation Support”, Pervasive Computing, Volume 11, Issue 4, pg 68-77, 2012

[18] S. Gandhi, A. Ganz, G. Mullett, “FIREGUIDE- Firefighter Guide and Tracker”, 32nd Annual International Conference of the IEEE, Aug. 31 – Sept. 4, 2010

[19] M. Scholz, T. Riedel, C. Decker, “A flexible architecture for a robust indoor navigation support device for firefighters”, Seventh International Conference on Networked Sensing Systems, 227-232, 15-18 Jun. 2010

[20] G. Liu, H. Tong, R. Zhang, “An Intelligent Control Architecture for Search Robot Based on Orthogonal Perception Information”, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2348 – 2352, 29-31 May 2012

[21] Y. Baudoin, D. Doroftei, G. De Cubber, S. Berrabah, “VIEW-FINDER: ROBOTICS ASSISTANCE TO FIRE-FIGHTING SERVICES AND CRISIS MANAGEMENT”, IEEE International Workshop on Security & Rescue Robotics, 1-6, 3-6 Nov. 2009

[22] M. Norouzi, A. Karambakhsh, M. Namazifar, B. Savkovic, “Object Based Navigation of Mobile Robot with Obstacle Avoidance using Fuzzy Controller”, IEEE International Conference on Control and Automation, 160-174, 9-11 Dec. 2009

[23] Polulu, Networks [Online], Available: http://www.polulu.com/

[24] Arduino, Networks [Online], Available: http://www.arduino.cc/

[25] Dagu Electronics, Networks [Online], Available: http://www.dagurobot.com/

[26] Wiki, Networks [Online], Available: http://en.wikipedia.org/wiki/Visibility_graph

[27] J. Xu; W. Liu; F. Lang; Y. Zhang; C. Wang, “Distance Measurement Model Based on RSSI in WSN”, Wireless Sensor Network, Vol. 2 Issue 8, p606, 2010

[28] A. Bose, Chuan Heng Foh, “A Practical Path Loss Model For Indoor WiFi Positioning Enhancement”, 6th International Conference on Information, Communications & Signal Processing, 1-5, 10-13 Dec., 2007

[29] Y. López, M. Gómez, J. Álvarez, F. Andrés, “Evaluation of an RSS-based Indoor Location System”, Sensors and Actuators A: Physical Volume 167, Issue 1, pg 110–116, 2011

[30] S. Farahani, “ZigBee Wireless Networks and Transceivers”, Newnes, 2008

[31] N. Gutierrez, C. Belmonte, J. Hanvey, R. Espejo, Z. Dong, “Indoor Localization for Mobile Devices”, 11th International Conference on Networking, Sensing and Control, 173-178, 7-9 Apr. 2014

[32] A. Bekkelien, “Bluetooth Indoor Positioning”, Master Thesis, University of Geneva, Mar. 2012

[33] B. Cook, G. Buckberry, I. Scowcroft, J. Mitchell, T. Allen, “Indoor Location Using Trilateration Characteristics”, London Communications Symposium, 8-9 September 2006

[34] S. Boonsriwai, A. Apavatjrut, “Indoor WIFI localization on Mobile Devices”, 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 1-5, 15-17 May, 2013

[35] Y. Wang, X. Yang, Y. Zhao, L. Cuthbert, “Bluetooth positioning using RSSI and triangulation methods”, Consumer Communications and Networking Conference, 837-842, 11-14 Jan. 2013

[36] D. Montemerlo, S. Thrun, W. Whittaker, “Conditional particle filters for simultaneous mobile robot localization and people-tracking”, IEEE International Conference on Robotics and Automation, 695-701 Vol.1, 11-15 May, 2002

[37] A. Raghavan, H. Ananthapadmanaban, M. Sivamurugan, B. Ravindran, “Accurate mobile robot localization in indoor environments using Bluetooth”, International Conference on Robotics and Automation, 4391-4396, 3-7 May, 2010

[38] O. Oguejiofor, V. Okorogu, A. Adewale, B. Osuesu, “Outdoor Localization System Using RSSI Measurement of Wireless Sensor Network”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2(2), January 2013

[39] D. Fox, W. Burgard, F. Dellaert, S., Thrun, “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots”, Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, 343-349, 18-22 July, 1999

[40] P. Hart, N. Nilsson, B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, IEEE Transaction on Systems Science and Cybernetics, 100-107, July, 1968

[41] A. Stentz, “Optimal and Efficient Path Planning for Partially-Known Environments”, Proceedings of the International Conference on Robotics and Automation, 3310–3317, May, 1994

[42] J. Park, H. Cho, S. Kim, D. Park, A. Kim, J. Park, “Adaptive Parameter Estimation Method for Wireless Localization Using RSSI Measurements”, 6th International Conference on Future Information Technology, 28-30 Jun, 2011

[43] K. Wendlandt, M. Khider, M. Angermann, R. Robertson, “Continuous location and direction estimation with multiple sensors using particle filtering”, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 92-97, Sept. 2006